HARDNESS AND TRACTABILITY FOR STRUCTURED NUMERICAL PROBLEMS

A Dissertation
Presented to
The Academic Faculty

By

Peng Zhang

In Partial Fulfillment
Of the Requirements for the Degree
Doctor of Philosophy in
Computer Science

Georgia Institute of Technology

December 2018

HARDNESS AND TRACTABILITY FOR STRUCTURED NUMERICAL PROBLEMS

Approved by:

Dr. Richard Peng, Advisor
School of Computer Science
Georgia Institute of Technology

Dr. Edmond Chow
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Petros Drineas
Department of Computer Science
Purdue University

Dr. Milena Mihail
School of Computer Science
Georgia Institute of Technology

Dr. Santosh Vempala
School of Computer Science
Georgia Institute of Technology

Date Approved: August 20 , 2018

To my parents.

ACKNOWLEDGEMENTS

First of all, I am grateful to my advisor, Richard Peng, for his support and patience during my PhD study at Georgia Tech. Richard introduced me to these interesting theories and problems in linear systems and linear programs. He also provided me many invaluable advices in both research and career.

Second, I would like to thank my thesis committee members: Edmond Chow, Petros Drineas, Milena Mihail, and Santosh Vempala. I also thank Eric Vigoda for being my qualifying exam committee member.

I am lucky to work with my co-author, Rasmus Kyng. Most of the works in this thesis were collaborated with him, from which I learnt a lot. I would like to thank Di Wang, for working together on positive LPs. I also enjoyed talking research with Anup Rao, who answered my various silly questions during my first year at GT.

I would like to thank Dan Spielman, for his kind help during my visit at Yale. Working with Dan is amazing, full of fun and surprises. Then, I would like to thank Gary Miller, for the wonderful summer at CMU. I really enjoyed talking about geometry and distributions. I also thank Yi Wu, for guiding me into numerical linear algebra and teaching me many basic things.

I thank Ahmed Sameh and Luca Dieci for teaching me linear algebra. I am also glad of being in classes taught by Santosh Vempala, Santanu Dey, and Robin Thomas.

My two and half years at Georgia Tech were great with friends: Zongchen Chen, Tung Mai, Samira Samadi, David Durfee, Saurabh Sawlani, Matthew Fahrbach, Sam Petti, Kevin Lai, Chunxing Yin, Zhuangdi Xu, Tianxin Tang, Shan Chen, and too many to mention. Besides, I would like to thank H. Venkateswaran and Dani Denton for their help on my PhD program. I also thank Richard Lipton for his books at theory lab.

Outside GT, I owe my thanks to people at Purdue: Elena Grigorescu, Greg Frederickson, Mikhail Atallah, and David Gleich, for their generous help during my two years there.

I also thank my colleagues and friends in theory group: Akash Kumar, Mingyuan Wang, Abram Magner, Abhiram Natarajan, G.V., Samson Zhou, and Young-San Lin.

I would like to thank people in China: Wei Chen, Xiaoming Sun, and Guochuan Zhang, for introducing me to theoretical computer science, and for their advices on research and career.

My special thanks go to Zichao Yang and Li Xiong, for talking and complaining during the five years in the US.

Last but not the least, I would like to thank my parents, for their lifetime support and love.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

We study structured linear systems and structured linear programs from both algorithm and complexity perspectives. These structured problems commonly arise in combinatorial optimization, machine learning, and operation research. Many of them can be solved significantly faster than general linear systems or linear programs by utilizing additional structures.

First, we consider linear systems in matrices which form a slightly larger class of graph Laplacians. Nearly-linear time algorithms have been designed for linear systems in graph Laplacians (Spielman-Teng, 2004), and sequentially for its generalizations, e.g., connection Laplacians (Kyng et al, 2016), and directed Laplacians (Cohen et al, 2017, 2018). In contrast to these positive results, we establish lower bounds for approximately solving linear systems in: multi-commodity Laplacian matrices, truss stiffness matrices, total variation matrices, all of which fall into the class of generalized graph Laplacians. Specifically, approximately solving linear systems in these matrices is equally hard as solving arbitrary linear systems. Our result answers a major question in this field (Spielman, 2016).

As a follow up, we prove hardness results for packing and covering LPs, which are special case of LPs with non-negative coefficients, constants and variables. We show that: any approximate solvers for packing and covering LPs whose run time has small dependence on error parameter would directly improve the best solvers for arbitrary linear systems. The latter is a long-standing open question in numerical linear algebra and scientific computing.

On the algorithmic side, we study linear systems in 3-D truss stiffness matrices with additional geometric structures. By combining Nested Dissection and support theory, which are both well-studied techniques for linear systems, we obtain an asymptotically faster algorithm. Moreover, we design a parallel algorithm for approximately solving mixed packing and covering LPs, which improves the algorithm by Young (2001).

CHAPTER 1

INTRODUCTION

A linear program (LP) is an optimization problem with linear constraints and linear objective function. In general, it can be written as:

$$\max_{\boldsymbol{x}}\{\boldsymbol{c}^\top \boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}, \tag{1.1}$$

where $\boldsymbol{A}$ is a given matrix, $\boldsymbol{b}, \boldsymbol{c}$ are given vectors, and $\boldsymbol{x}$ is an unknown vector. There are many other equivalent forms of linear programs. A linear system is a system of linear equations. Its input consists of a coefficient matrix $\boldsymbol{A}$, a column vector $\boldsymbol{b}$, and its output is a vector $\boldsymbol{x}$ such that[1]

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}.$$

In practice, we allow small error $\epsilon > 0$, and then the goal becomes to compute an approximate vector $\tilde{\boldsymbol{x}}$ such that

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_2 \leq \epsilon \|\boldsymbol{b}\|_2 .$$

There are several different definitions of approximately solving an LP, which we will not discuss in this thesis. We suggest readers to refer to [1, 2].

Linear programs and linear systems are central tools in computer science, combinatorial optimization, statistics, economics, and engineering. A variety of combinatorial problems, such as network flow problems, multicommodity flow problems, scheduling problems, etc., can be formulated as linear programs. One widely studied problem for modeling in statistics is linear regression problem. Given a (tall) matrix $\boldsymbol{A}$, a column vector $\boldsymbol{b}$, it seeks a vector $\boldsymbol{x}$ to minimize $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p$ for some norm p. When $p = 1$ or ∞, this problem is

[1] Here we assume $\boldsymbol{A}$ is invertible, we will discuss more general cases in Section 2.2.

equivalent to solving a linear program; when $p = 2$, this problem is equivalent to solving a linear system, which is usually referred to as the normal equation. Besides, both linear programs and linear systems are frequently used as sub-routines for other optimization problems.

Linear programs and linear systems are related via Interior-Point Method, which solves a linear program in provably polynomial time [3]. Interior-point method reduces solving a linear program of the form (1.1) to solving a sequence of $\widetilde{O}(\sqrt{\text{rank}(\boldsymbol{A})}L)$[2] linear systems,where L is the bit complexity of the input [4]. Most combinatorial problems have polynomially bounded input numbers, and thus $L = O(\log n)$ is ignorable. Thus, any improvement on solving linear systems would imply a direct improvement on solving linear programs.

Interior-point method has advantages over other LP algorithms either in practice or in theory. Ellipsoid method is the first LP algorithm which was proved to have polynomial running time [5]. However, it has high cost in practice. The simplex algorithm runs well in practice, but there exists an example showing that the simplex algorithm needs exponential running time in the worst case [6]. About twenty years ago, it was shown that the simplex algorithm has smoothed polynomial time [7], which explains its good performance in practice.

In general, algorithms for solving linear programs or linear systems have high complexity in the worst case[3]. Asymptotically faster algorithms (even nearly-linear time algorithms) exist when inputs have some additional structures.

In this thesis, we study structured linear systems and linear programs from both algorithm and complexity perspectives. We consider linear systems in matrices which form a slightly larger class of graph Laplacians, referred to as Graph-Structured Block Matrices (GSBMs) [8]. In contrast to the existing nearly-linear time solvers for graph Laplacians [9] and its generalizations [10, 11, 12], we prove hardness results for GSBMs: approximately

[2]We use $\widetilde{O}(\cdot)$ to hide log factors.

[3]It is open whether linear programs are solvable in strongly-polynomial time.

solving linear systems in GSBMs is equally hard as approximately solving arbitrary linear systems. Building upon linear system based hardness assumptions, we then establish conditional lower bounds for packing and covering LPs, which are a special case of LPs with non-negative coefficients, constants and variables.

On the algorithmic side, we obtain an asymptotically faster solver for linear systems in 3-D trusses with additional geometric structures. General truss matrices are GSBMs. Moreover, we design a parallel algorithm for approximately solving mixed packing and covering LPs, which improves the algorithm by Young [13].

Solving (Arbitrary) Linear Systems. Given a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, one can directly solve it by Gaussian elimination, which has running time $O(n^3)$. Thanks to the fast matrix multiplication since Strassen [14], one can speed up solving the linear system by inverting the coefficient matrix $\boldsymbol{A}$ and then multiplying $\boldsymbol{A}^{-1}\boldsymbol{b}$. It was shown that one can invert an $n \times n$ nonsingular matrix by recursively applying matrix multiplication, which takes $O(n^\omega)$ time [15]. Here, $\omega < 2.3728639$ [14, 16, 17] is the matrix multiplication constant. Using a similar trick, one can multiply two matrices by computing matrix inversions $\log$ times. Thus, matrix multiplication and matrix inversion have same complexity up to a $\log$ factor.

Note that $\boldsymbol{A}^{-1}$ can be dense, even if $\boldsymbol{A}$ itself is sparse. Thus, the matrix multiplication constant $\omega \geq 2$. Raz showed that multiplying two $n \times n$ matrices needs time $\Omega(n^2 \log n)$ [18]. There is no other non-trivial lower bound for ω.

This simple algorithm by fast matrix inversion is the best known algorithm for solving an arbitrary linear system. It is not known whether one can solve an arbitrary linear system in $n \times n$ dimensions in time $o(n^\omega)$.

When a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is sparse (e.g., $\boldsymbol{A}$ has $n \log n$ non-zeros), it is desirable to solve this linear system in time slightly larger than $\mathrm{nnz}(\boldsymbol{A})$, the number of non-zeros of $\boldsymbol{A}$[4]. Iterative methods aim at this goal, by avoiding matrix inversion. The operations of

[4]One needs to read all entries of $\boldsymbol{A}$ to solve a linear system with small error.

each iteration only include matrix-vector multiplication and vector-vector operations. The best known iterative method for solving an arbitrary linear system is Conjugate Gradient Descent or its variants [19] (e.g., Chebyshev Iterations, or an accelerated first order method such as Nesterov's algorithm). These methods solve a linear system whose coefficient matrix $\boldsymbol{A}$ is symmetric and PSD in time $O(\mathrm{nnz}(\boldsymbol{A})\sqrt{\kappa(\boldsymbol{A})}\log(1/\epsilon))$, where $\kappa(\boldsymbol{A})$ is the condition number of $\boldsymbol{A}$.

Conjugate gradient can also be shown in the RealRAM model to converge to an exact solution in n iterations, giving a running time of order $n \cdot \mathrm{nnz}(\boldsymbol{A})$. However, this is highly misleading, because achieving this type of behavior in a fixed point arithmetic model requires about n bits per number, as opposed to $\mathrm{poly}\log(n/\epsilon)$ bits to achieve the condition number dependent behavior [20].

Graph Laplacians, its Generalizations, and Nearly-Linear Time Solvers. Graph Laplacians are a class of well-studied structured matrices. Linear systems in graph Laplacians can be solved in nearly-linear time.

An $n \times n$ Laplacian matrix can be viewed as an undirected graph over n vertices. Each vertex of the graph corresponds to a single row and a single column of the matrix. Each edge corresponds to an n-dimensional vector whose non-zeros are only at the position of its two endpoints. Taking the outer product of the edge vector, we get a rank-one matrix. The Laplacian matrix is then defined as a sum over these edge rank-one matrices.

Laplacian linear systems arise from using interior-point methods for combinatorial problems on graphs. For example, max flow in directed unweighted graphs [21, 22], negative weight shortest paths and max weight matchings [23], min cost flows and lossy generalized flows [24, 4], etc. Using fast Laplacian solvers to speeding up graph algorithms was formulated as the Laplacian Paradigm in [25].

Spielman and Teng designed the first *nearly-linear time* solver for Laplacian linear systems [9]. Specifically, their algorithms solves a linear system in a Laplacian matrix with

m nonzeros up to accuracy $\epsilon > 0$ in time $O(m \log^{O(1)} n \log(1/\epsilon))$. The exponent constant of $\log n$ was then improved by a sequence of following works (see [26, 27, 28, 29, 30, 31], etc.).

A natural generalization of graph Laplacians is to allow *multiple* labels for each vertex. These matrices arise from using interior-point methods solving more generalized graph problems, e.g., multi-commodity flow problems [21, 22], and total variation minimization problem in images [32]. In addition, these matrices have been studied in Markov random fields [33], image processing [34, 35, 36], Euclidean embedding of graphs [37], data processing for cryo-electron microscopy [38, 39, 40], phase retrieval [41, 42], and linear elasticity [43, 44].

Connection Laplacians are one example, for which nearly-linear time solvers exist [10]. These matrices have applications in image and signal processing [45, 46]. Other examples include Truss stiffness matrices [43, 44], multi-commodity Laplacians [21, 22], and total variation minimization matrices [32]. However, it is not known whether one can solve a linear system in one of these matrices in nearly-linear time.

Geometric Structures and Nested Dissection. Another widely studied class of structured linear systems, for which asymptotically faster algorithms exist, is linear systems whose coefficient matrices' non-zeros have geometrically embedded structures. Given an $n \times n$ matrix $\boldsymbol{A}$, its non-zero structures can be viewed as a graph over n vertices: There is an edge between vertices i and j iff the (i, j)th entry or the (j, i)th entry of $\boldsymbol{A}$ is non-zero. We call this graph as G_A.

Applying Gaussian elimination to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ can be viewed as operations on G_A: Eliminate the ith row and the ith column of $\boldsymbol{A}$ corresponds to deleting the vertex i of G_A and putting a clique over the neighbors of i. This process may introduce new non-zeros in the matrix (correspondingly, new edges on the graph), which we refer to as fill-ins. The size of fill-ins directly determines the running time of Gaussian elimination. Even if $\boldsymbol{A}$ itself

is sparse, the intermediate matrix produced by partial Gaussian elimination can be dense. The key to speeding up Gaussian elimination is to finding an elimination ordering which produces a small number of fill-ins.

Nested Dissection is used to compute an ordering for sparse Gaussian elimination when matrices $\boldsymbol{A}$ with additional geometric structures, for example, G_A is a planar graph, a well-shaped 3-D tetrahedral meshes, or a graph without some minors [47, 48, 49, 50]. These graphs have small separators, whose removal divides a graph into balanced pieces. Nested dissection recursively computes small separators of G_A, and then eliminate vertices in a reverse order. Using this idea, one can solve a linear system in a 2-D planar in time $O(n^{1.5})$ [48], and a linear system in a well-shaped 3-D tetrahedral mesh in time $O(n^2)$ [49].

Packing and Covering LPs. Packing and covering LPs are LPs formulated with non-negative coefficients, constants and variables. These LPs commonly arise from theoretical computer science, operation research, and optimization, etc. Moreover, they can be approximately solved in nearly linear time.

Formally, a packing LP has its generic form

$$\max_{\boldsymbol{x}\geq\boldsymbol{0}}\{\boldsymbol{c}^\top\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x}\leq\boldsymbol{b}\},$$

where $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$ have non-negative entries. A covering LP is of the form

$$\min_{\boldsymbol{x}\geq\boldsymbol{0}}\{\boldsymbol{c}^\top\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x}\geq\boldsymbol{b}\},$$

where $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$ again are entrywise non-negative. One more general class is mixed packing and covering LPs, which is sometimes simply referred to as positive LPs. A positive LP in general has both packing constraints and covering constraints.

Note that a packing LP or a covering LP is always feasible, and has non-negative optimal value due to the all-zero solution. Besides, a packing LP and a covering LP form

a primal-dual pair. Let OPT denote the optimal value of this primal-dual pair. We say $\boldsymbol{x}$ is a $(1-\epsilon)$-approximation for the packing LP if $\boldsymbol{x}$ is feasible and $\boldsymbol{c}^\top \boldsymbol{x} \geq (1-\epsilon)\text{OPT}$. Similarly, we say $\boldsymbol{x}$ is a $(1+\epsilon)$-approximation for the covering LP if $\boldsymbol{x}$ is feasible and $\boldsymbol{c}^\top \boldsymbol{x} \leq (1+\epsilon)\text{OPT}$.

We are interested in *nearly-liner time* algorithms for approximately solving packing and covering LPs. Specifically, the running time is linear in the number of input non-zeros or dimensions, and polynomial in $1/\epsilon$. Note the running time is independent of input coefficients, and thus these algorithms are referred to as width-independent algorithms in the literature.

Although one can use ellipsoid methods or interior-point methods to get $\log(1/\epsilon)$-dependence [5, 3], these algorithms usually have large dependence on the dimension and the bit-complexity of inputs, which are not desirable in practice. The nearly-linear time solvers for packing and covering LPs are mainly based on first-order methods, e.g., gradient descent, multiplicative weights update [51, 13, 52, 53, 54, 55].

1.1 Our Results

Hardness Results for Graph-Structured Linear Systems. We consider three types of Graph-Structured Block Matrices (GSBM): Multi-commodity Laplacians, Truss Stiffness Matrices, and Total Variation Matrices. All these matrices are generalizations of graph Laplacians by allowing multiple labels for each vertex. We are interested in solving linear systems in GSBMs such that the running time has logarithmic dependence on condition number of coefficient matrices and accuracy. This is the kind of running time dependence established for graph Laplacians, Connection Laplacians, directed Laplacians, and planar 2-D truss stiffness matrices. The following result is joint with Kyng [56].

Theorem 1.1.1 (Hardness for Graph-Structured Linear Systems)**.** *We consider three types of Graph-Structured Block Matrices: 2-commodity Laplacians, 2-D Truss Stiffness Matrices, and 2-Total Variation Matrices. Suppose that for one or more of these classes, the*

linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ in a matrix $\boldsymbol{A}$ with s non-zeros can be solved in time $\widetilde{O}(s^a)$, for some constant $a \geq 1$, with the running time having logarithmic dependence on condition number and accuracy.Then linear systems in all matrices *with polynomially bounded integer entries and condition number can be solved to high accuracy in time $\widetilde{O}(s^a)$, where again s is the number of non-zero entries of the matrix.*

Particularly, if one can solve a linear system in a 2-commodity Laplacian, or a 2-D truss stiffness matrix, or a 2-total variation matrix in nearly linear time, as the existing solvers for graph Laplacians and connection Laplacians, then one can solve an arbitrary linear system with bounded entries and condition number in nearly linear time. However, the best known algorithm for solving an arbitrary linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ of dimension $n \times n$ runs in time $O(\min\{n^\omega, \text{nnz}(\boldsymbol{A})\sqrt{\kappa(\boldsymbol{A})}\log(1/\epsilon)\})$[5]. This is asymptotically larger than any nearly linear function of $\text{nnz}(\boldsymbol{A})$, for $\text{nnz}(\boldsymbol{A}) \ll n^2$.

Hardness Results for Packing LPs. We have two hardness results for packing LPs. One concerns a dense packing LP, for which the running time is usually a function of dimensions. The other one is related to a sparse packing LP, for which the running time is a linear function of the number of non-zeros. Since packing LPs and covering LPs are primal and dual to each other, our results also hold for covering LPs. Both Theorem 1.1.2 and 1.1.3 are joint with Kyng, Peng and Di [57].

Theorem 1.1.2. *If one can $(1-\epsilon)$-approximately solve a packing LP of dimension $O(n) \times O(n)$ in time $O(n^2/\epsilon^{0.0009})$, then one can solve an arbitrary linear system of dimension $O(n) \times O(n)$ asymptotically faster than multiplying two $n \times n$ matrices.*

As mentioned earlier in Introduction, the best known algorithm for solving an arbitrary linear system of dimension $n \times n$ runs in time $O(n^\omega)$, for $\omega < 2.373$. If ω is strictly larger than 2 (e.g., $\omega > 2.00001$), then Theorem 1.1.2 would imply that "matrix inversion-by-matrix multiplication" is suboptimal to solving an arbitrary linear system.

[5] Or $\text{nnz}(\boldsymbol{A})\kappa(\boldsymbol{A})\log(1/\epsilon)$ if $\boldsymbol{A}$ is not PSD.

Our second result is related to sparse systems.

Theorem 1.1.3. *If one can $(1-\epsilon)$-approximately solve a packing LP with N non-zeros in time $O(N/\epsilon^{0.22})$, then one can solve an arbitrary linear system asymptotically faster than applying Conjugate Gradient.*

Conjugate Gradient (CG) is the best known algorithm for solving a sparse and well-conditioned linear system. The running of applying CG to a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with $\boldsymbol{A}$ being symmetric and PSD is $O(\mathrm{nnz}(\boldsymbol{A})\sqrt{\kappa(\boldsymbol{A})}\log(1/\epsilon))$, where $\mathrm{nnz}(\boldsymbol{A})$ is the number of non-zeros of $\boldsymbol{A}$, $\kappa(\boldsymbol{A})$ is the condition number, and ϵ is the accuracy parameter. When $\mathrm{nnz}(\boldsymbol{A}) \approx n \log n$ and $\kappa(\boldsymbol{A}) < n^{1.99}$, CG is faster than directly inverting $\boldsymbol{A}$, even if the matrix multiplication constant $\omega = 2$.

Note for an arbitrary linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ satisfying the following: $\dim(\boldsymbol{A}) = n \times n$, $\mathrm{nnz}(\boldsymbol{A}) = O(n \log n)$, and $\kappa(\boldsymbol{A}) = n^2$, either fast matrix inversion or CG takes time $\Omega(n^2)$, given the fact that $\omega \geq 2$. It is not known whether one can approximately solve such a linear system in time $o(n^2)$. By Theorem 1.1.3, a fast packing LP solver would imply a positive answer to this question.

Trevisan and Xhafa [58] studied parallel complexity of packing LPs, and they showed that exactly solving a packing LP is P-complete. However, there is no known hardness result for approximately solving a packing LP. In [58], the authors gave an example showing that their reduction does not work for approximate solvers.

Algorithms for Truss Linear Systems with Geometric Structures. As stated in Theorem 1.1.1, solving an arbitrary 3-D truss linear system[6] is as hard as solving an arbitrary linear system. We consider 3-D trusses with additional geometric structures, which commonly arise from scientific computing.

A 3D truss is said to be *simple* iff it is a simplicial complex and every tetrahedron has bounded aspect ratio. We say a 3D truss is *edge-simple* iff its tetrahedral mesh is simple

[6] 2-D truss is a special case of 3-D, by fixing one of the three dimensions to be 0.

and every tetrahedron has bounded edge lengths and stiffness coefficients (i.e., both are bounded above and below by constants). The following two results are from the paper [59] coauthored with Kyng, Peng and Schwieterman.

Theorem 1.1.4. *Consider an edge-simple 3-D truss $\mathcal{T}$ over n vertices, which is formed by a union of k convex simplicial complexes with* constant *aspect ratio each. There is an algorithm which solves a linear system in the stiffness matrix of $\mathcal{T}$ up to accuracy $\epsilon > 0$ in time $O(k^{1/3}n^{5/3}\log(1/\epsilon))$*[7].

Theorem 1.1.5. *Consider an edge-simple 3-D truss $\mathcal{T}$ over n vertices, which is formed by a union of k convex simplicial complexes with* arbitrary *aspect ratio each. There is an algorithm which solves a linear system in the stiffness matrix of $\mathcal{T}$ up to accuracy $\epsilon > 0$ in time $O(k^{22/3}n^{11/6}\log(1/\epsilon))$.*

Theorem 1.1.4 concerns a union of k convex truss with small aspect ratios, which improves the $O(n^2)$ running time by nested dissection [49] for all $k \ll n$. Theorem 1.1.5 allows truss components to have arbitrary aspect ratios, which improves on nested dissection provided $k \ll n^{1/44}$.

Our requirements for trusses have more restrictions than the previous nested dissection based algorithms by Miller and Thurston [49], which only requires constant aspect ratios for individual tetrahedrons. Constant aspect ratios are crucial for nested dissection. Miller and Thurston provided a mesh of tetrahedrons with arbitrary aspect ratios, for which no sub-linear size separator exists and thus nested dissection cannot speed up Gaussian elimination. Our additional restrictions are mainly due to the need to derive spectral bounds for the preconditioner. Assumptions such as bounded edge lengths and stiffness coefficients (in addition to aspect ratios) are also present in previous work by Daitch and Spielman [44], which forms the starting point for our key technical results on eigenvalues of 3-D simplicial complexes. Furthermore, the need to maintain null spaces as well as truss structures

[7] In both Theorem 1.1.4 and 1.1.5, we assume matrix multiplication constant $\omega = 3$, following the convention of nested dissection community. We will discuss the running times in term of the best known $\omega < 2.3728639$ in Section 5.6.

in intermediate steps of Gaussian elimination places additional requirements on the mesh structure (mainly convexity).

We remark that the geometric assumptions of Theorem 1.1.5 are in many ways fairly weak. We need the individual truss tetrahedrons to have small aspect ratio, but each of the k truss components may overall have a wide range of shapes: It can form a ball, a pancake, or even a very long beam with arbitrarily large aspect ratio. The dependence on k is fairly bad and has not been carefully optimized, meaning currently that only about $k \ll n^{1/44} \approx n^{0.0227}$ convex edge-simple trusses can be combined while still achieving a speed-up over nested dissection. Even so, this allows the construction of some shapes with genus up to $n^{1/44}$.

Parallel Algorithm for Positive LPs. We design the currently fast algorithm for approximately solving positive LPs in parallel. We use the following form of a positive LP:

$$\min_{\boldsymbol{x} \geq \mathbf{0}} \{\lambda : \boldsymbol{P}\boldsymbol{x} \leq \lambda \boldsymbol{p}, \boldsymbol{C}\boldsymbol{x} \geq \boldsymbol{c}\},$$

where $\boldsymbol{P}, \boldsymbol{C}, \boldsymbol{p}, \boldsymbol{c}$ are all entrywise non-negative. A pair $(\lambda, \boldsymbol{x})$ is a $(1+\epsilon)$-approximation if it is a feasible solution and $\lambda \leq (1+\epsilon)\text{OPT}$, where OPT is the optimal value of the above LP. The following result is a joint work with Mahoney, Rao and Wang [60].

Theorem 1.1.6. *Given a positive LP with N non-zero entries, there exists a parallel algorithm outputs a $(1+\epsilon)$-approximate solution with running time $\widetilde{O}(1/\epsilon^3)$ and total work $\widetilde{O}(N/\epsilon^3)$.*

This result improves Young's algorithm in 2001 [13] by a factor $1/\epsilon$. Our algorithm follows the Lagrangian-relaxation approach, and can be viewed as a modification of Young's algorithm. The bottleneck of each iteration is matrix-vector multiplication, which can be implemented parallelly in $O(\log N)$ depth. Thus the running time is the number of iterations up to a log factor. The total work counts the number of total operations of the

algorithm. In general, the running time (or the number of iterations) is more interesting than the total work.

Recall that pure packing or covering LPs are a special case of positive LPs. It turns out that our algorithm approximately solves a pure packing or covering LP in $\widetilde{O}(1/\epsilon^2)$ iterations. This matches the best known results in [54, 61], while our algorithm is deterministic and has simpler analysis.

1.2 Organization of This Dissertation

We give formal notations and definitions, which will be used throughout this thesis, in Chapter 2. We prove Theorem 1.1.1 in Chapter 3, Theorem 1.1.2 and 1.1.3 in Chapter 4, Theorem 1.1.4 and Theorem 1.1.5 in Chapter 5, and Theorem 1.1.6 in Chapter 6.

CHAPTER 2

PRELIMINARIES

2.1 Vectors and Matrices

Indexing. We use subscripts to denote entries of a matrix or a vector. Given a vector $\boldsymbol{x} \in \mathbb{R}^n$, for $1 \leq i < i+j \leq n$, we denote $\boldsymbol{x}_i$ the ith entry of $\boldsymbol{x}$, and we denote $\boldsymbol{x}_{i:i+j}$ the subvector whose entries are $\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_{i+j}$. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, let $\boldsymbol{A}_i$ denote the ith row of $\boldsymbol{A}$, and let $\boldsymbol{A}_{ij}$ denote the (i,j)th entry of $\boldsymbol{A}$. We use superscripts to index a sequence of matrices or vectors, e.g., $\boldsymbol{A}^1, \boldsymbol{A}^2, \ldots,$ and $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$, except when some other meaning is clearly stated. In addition, we use $\mathrm{nnz}(\boldsymbol{A})$ to denote the number of nonzero entries of $\boldsymbol{A}$, and we use $\dim(\boldsymbol{A})$ to denote the dimensions of $\boldsymbol{A}$.

Spectral definitions. A symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is *positive semi-definite* (PSD) iff $\boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} \geq 0$ holds for every $\boldsymbol{x} \in \mathbb{R}^n$. For any matrix $\boldsymbol{B} \in \mathbb{R}^{m \times n}$, we denote $\lambda_{\max}(\boldsymbol{B})$ the largest eigenvalue of $\boldsymbol{B}$, and $\lambda_{\min}(\boldsymbol{B})$ the smallest *nonzero* eigenvalue of $\boldsymbol{B}$. Let $\sigma_{\max}(\boldsymbol{B})$ be the largest singular value of $\boldsymbol{B}$, and $\sigma_{\min}(\boldsymbol{B})$ be the smallest *nonzero* singular value of $\boldsymbol{B}$. If $\boldsymbol{B}$ is a symmetric PSD matrix, then the eigenvalues of $\boldsymbol{B}$ and the singular values of $\boldsymbol{B}$ coincide. Let $\kappa(\boldsymbol{B}) \stackrel{\text{def}}{=} \sigma_{\max}(\boldsymbol{B}) / \sigma_{\min}(\boldsymbol{B})$ denote the condition number of $\boldsymbol{B}$.

For two symmetric matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$, we say $\boldsymbol{A} \succcurlyeq \boldsymbol{B}$ iff $\boldsymbol{A} - \boldsymbol{B}$ is PSD. The condition number of $\boldsymbol{A}$ relative to $\boldsymbol{B}$, denoted by $\kappa(\boldsymbol{A}, \boldsymbol{B})$, to be

$$\min \left\{ \frac{\lambda_{\max}}{\lambda_{\min}} : \lambda_{\min} \cdot \boldsymbol{B} \preccurlyeq \boldsymbol{A} \preccurlyeq \lambda_{\max} \cdot \boldsymbol{B} \right\}.$$

Consider the eigen-decomposition of a PSD rank-k matrix $\boldsymbol{A} = \sum_{1 \leq i \leq k} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^\top$, the *Moore-Penrose pseudo-inverse* of $\boldsymbol{A}$ is $\boldsymbol{A}^\dagger \stackrel{\text{def}}{=} \sum_{1 \leq i \leq k} \lambda_i^{-1} \boldsymbol{u}_i \boldsymbol{u}_i^\top$.

Given a matrix $\boldsymbol{A}$, let $\mathrm{im}(\boldsymbol{A})$ to denote the image of a matrix $\boldsymbol{A}$. Let $\boldsymbol{\Pi}_{\boldsymbol{A}} \stackrel{\text{def}}{=} \boldsymbol{A}(\boldsymbol{A}\boldsymbol{A}^\top)^\dagger \boldsymbol{A}^\top$,

i.e. the orthogonal projection onto $\mathrm{im}(\boldsymbol{A})$. Note that $\boldsymbol{\Pi}_{\boldsymbol{A}} = \boldsymbol{\Pi}_{\boldsymbol{A}}^\top$ and $\boldsymbol{\Pi}_{\boldsymbol{A}} = \boldsymbol{\Pi}_{\boldsymbol{A}}^2$.

Norms. Given a vector $\boldsymbol{x} \in \mathbb{R}^n$, the Euclidean norm (or ℓ_2 norm) is defined as $\|\boldsymbol{x}\|_2 \stackrel{\text{def}}{=} \sqrt{\sum_{1\le i\le n} \boldsymbol{x}_i^2}$. Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a symmetric PSD matrix, we define the matrix norm w.r.t. $\boldsymbol{A}$ of a vector $\boldsymbol{x}$ as $\|\boldsymbol{x}\|_{\boldsymbol{A}} \stackrel{\text{def}}{=} \sqrt{\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}}$. For matrix $\boldsymbol{A}$, we define

$$\begin{aligned}
\|\boldsymbol{A}\|_{\max} &\stackrel{\text{def}}{=} \max_{i,j} |\boldsymbol{A}_{ij}|\,, \\
\|\boldsymbol{A}\|_1 &\stackrel{\text{def}}{=} \max_j \sum_i |\boldsymbol{A}_{ij}|\,, \\
\|\boldsymbol{A}\|_\infty &\stackrel{\text{def}}{=} \max_i \sum_j |\boldsymbol{A}_{ij}|\,, \\
\|\boldsymbol{A}\|_2 &\stackrel{\text{def}}{=} \sqrt{\lambda_{\max}(\boldsymbol{A}\boldsymbol{A}^\top)}.
\end{aligned}$$

We let $\min_+(\boldsymbol{A}) = \min_{i,j \text{ s.t. } \boldsymbol{A}_{ij}\neq 0} |\boldsymbol{A}_{ij}|$.

Schur complement. Schur complement is an intermediate matrix of partial Gaussian elimination.

Definition 2.1.1 (Schur complement)**.** *Let S, T be a partition of the indices of a square matrix $\boldsymbol{A}$ so that*

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{SS} & \boldsymbol{A}_{ST} \\ \boldsymbol{A}_{ST}^\top & \boldsymbol{A}_{TT} \end{pmatrix}, \tag{2.1}$$

where $\boldsymbol{A}_{SS}, \boldsymbol{A}_{ST}, \boldsymbol{A}_{TT}$ are block matrices, the Schur complement *of $\boldsymbol{A}$ onto T is*

$$\textsc{Sc}[\boldsymbol{A}]_T \stackrel{\text{def}}{=} \boldsymbol{A}_{TT} - \boldsymbol{A}_{ST}^\top \boldsymbol{A}_{SS}^{-1} \boldsymbol{A}_{ST}.$$

The following are useful facts of Schur complements, which we will prove in Appendix A.1.

Fact 2.1.2. *Consider a matrix $\boldsymbol{A}$ defined as in Equation* (2.1)*, and any fixed vector $\boldsymbol{x}$ of dimension $|S|$. Then,*

$$\min_{\boldsymbol{y}\in\mathbb{R}^{|T|}} \begin{pmatrix} \boldsymbol{x}^\top & \boldsymbol{y}^\top \end{pmatrix} \boldsymbol{A} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \boldsymbol{x}^\top \mathrm{SC}[\boldsymbol{A}]_T \, \boldsymbol{x}.$$

Fact 2.1.3. *Let $\boldsymbol{A}$ be a symmetric PSD matrix and $\mathrm{SC}[\boldsymbol{A}]_T$ be its Schur complement.*

1. *$\mathrm{SC}[\boldsymbol{A}]_T$ is a symmetric PSD matrix.*
2. *$\lambda_{\max}(\mathrm{SC}[\boldsymbol{A}]_T) \leq \lambda_{\max}(\boldsymbol{A})$.*

2.2 Approximately Solving A Linear System

In this section we formally define the notions of approximate solutions to linear systems.

Definition 2.2.1 (Linear System Approximation Problem (LSA))**.** *Given linear system $(\boldsymbol{A}, \boldsymbol{c})$, where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, and $\boldsymbol{c} \in \mathbb{R}^m$, and given a scalar $0 \leq \epsilon \leq 1$, we refer to the* LSA *problem for the triple $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$ as the problem of finding $\boldsymbol{x} \in \mathbb{R}^n$ s.t.*

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}\|_2 \leq \epsilon \, \|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}\|_2 \,,$$

and we say that such an $\boldsymbol{x}$ is a solution to the LSA *instance $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$.*

This definition of a LSA instance and solution has several advantages: when $\mathrm{im}(\boldsymbol{A}) = \mathbb{R}^m$, we get $\boldsymbol{\Pi}_{\boldsymbol{A}} = \boldsymbol{I}$, and it reduces to the natural condition $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{c}\|_2 \leq \epsilon \, \|\boldsymbol{c}\|_2$, which because $\mathrm{im}(\boldsymbol{A}) = \mathbb{R}^m$, can be satisfied for any ϵ, and for $\epsilon = 0$ tells us that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}$.

When $\mathrm{im}(\boldsymbol{A})$ does not include all of $\mathbb{R}^m$, the vector $\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}$ is exactly the projection of $\boldsymbol{c}$ onto $\mathrm{im}(\boldsymbol{A})$, and so a solution can still be obtained for any ϵ. Further, as $(\boldsymbol{I} - \boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{c}$ is

orthogonal to $\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}$ and $\boldsymbol{A}\boldsymbol{x}$, it follows that

$$\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{c}\|_2^2 = \|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{c}\|_2^2 + \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}\|_2^2 .$$

Thus, when $\boldsymbol{x}$ is a solution to the LSA instance $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$, then $\boldsymbol{x}$ also gives an $\epsilon^2 \|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}\|_2^2$ additive approximation to

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{c}\|_2^2 = \|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{c}\|_2^2 . \tag{2.2}$$

Similarly, an $\boldsymbol{x}$ which gives an additive $\epsilon^2 \|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}\|_2^2$ approximation to Problem (2.2) is always a solution to the LSA instance $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$. These observations prove the following (well-known) fact:

Fact 2.2.2. *Let $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathbb{R}^m} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{c}\|_2^2$, then for every $\boldsymbol{x}$,*

$$\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{c}\|_2^2 \le \|\boldsymbol{A}\boldsymbol{x}^*-\boldsymbol{c}\|_2^2 + \epsilon^2 \|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}\|_2^2$$

if and only if $\boldsymbol{x}$ is a solution to the LSA *instance $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$.*

When the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}$ does not have a solution, a natural notion of solution is any minimizer of Problem (2.2). A simple calculation shows that this is equivalent to requiring that $\boldsymbol{x}$ is a solution to the linear system $\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^\top\boldsymbol{c}$, which always has a solution even when $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}$ does not. The system $\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^\top\boldsymbol{c}$ is referred to as the *normal equation* associated with $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}$ (see [62]).

Fact 2.2.3. *$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{c}\|_2^2$, if and only if $\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{A}^\top\boldsymbol{c}$, and this linear system always has a solution.*

This leads to a natural question: Suppose we want to approximately solve the linear system $\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^\top\boldsymbol{c}$. Can we choose our notion of approximation to be equivalent to that of a solution to the LSA instance $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$?

A second natural question is whether we can choose a notion of distance between a proposed solution $\boldsymbol{x}$ and an optimal solution $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{c}\|_2^2$ s.t. this distance being small is equivalent to $\boldsymbol{x}$ being a solution to the LSA instance $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$? The answer to both questions is yes, as demonstrated by the following facts:

Fact 2.2.4. *Suppose* $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{c}\|_2^2$ *then*

1. $\left\|\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{c}\right\|_{(\boldsymbol{A}^\top\boldsymbol{A})^\dagger} = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}\|_2 = \|\boldsymbol{x} - \boldsymbol{x}^*\|_{\boldsymbol{A}^\top\boldsymbol{A}}$.
2. *The following statements are each equivalent to* $\boldsymbol{x}$ *being a solution to the* LSA *instance* $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$*:*
   (a) $\left\|\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{c}\right\|_{(\boldsymbol{A}^\top\boldsymbol{A})^\dagger} \leq \epsilon \left\|\boldsymbol{A}^\top\boldsymbol{c}\right\|_{(\boldsymbol{A}^\top\boldsymbol{A})^\dagger}$ *if and only if* $\boldsymbol{x}$ *is a solution to the* LSA *instance* $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$.
   (b) $\|\boldsymbol{x} - \boldsymbol{x}^*\|_{\boldsymbol{A}^\top\boldsymbol{A}} \leq \epsilon \|\boldsymbol{x}^*\|_{\boldsymbol{A}^\top\boldsymbol{A}}$ *if and only if* $\boldsymbol{x}$ *is a solution to the* LSA *instance* $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$.

For completeness, we prove Fact 2.2.4 in Appendix A.2. Fact 2.2.4 explains connection between our Definition 2.2.1, and the usual convention for measuring error in the Laplacian solver literature [9]. In this setting, we consider a Laplacian matrix $\boldsymbol{L}$, which can be written as $\boldsymbol{L} = \boldsymbol{A}^\top\boldsymbol{A} \in \mathbb{R}^{n\times n}$, and a vector $\boldsymbol{b}$ s.t. $\boldsymbol{\Pi}_{\boldsymbol{A}^\top\boldsymbol{A}}\boldsymbol{b} = \boldsymbol{b}$. This condition on $\boldsymbol{b}$ is easy to verify in the case of Laplacians, since for the Laplacian of a connected graph, $\boldsymbol{\Pi}_{\boldsymbol{A}^\top\boldsymbol{A}} = \boldsymbol{I} - \frac{1}{n}\boldsymbol{1}\boldsymbol{1}^\top$. Additionally, it is also equivalent to the condition that there exists $\boldsymbol{c}$ s.t. $\boldsymbol{b} = \boldsymbol{A}^\top\boldsymbol{c}$. For Laplacians it is possible to compute both $\boldsymbol{A}$ and a vector $\boldsymbol{c}$ s.t. $\boldsymbol{b} = \boldsymbol{A}^\top\boldsymbol{c}$ in time linear in $\mathrm{nnz}(\boldsymbol{L})$. For Laplacian solvers, the approximation error of an approximate solution $\boldsymbol{x}$ is measured by the ϵ s.t. $\left\|\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\right\|_{(\boldsymbol{A}^\top\boldsymbol{A})^\dagger} \leq \epsilon \|\boldsymbol{b}\|_{(\boldsymbol{A}^\top\boldsymbol{A})^\dagger}$. By Fact 2.2.4, we see that this is exactly equivalent to $\boldsymbol{x}$ being a solution to the LSA instance $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$.

The following Lemma states that, if $\Pi_A\boldsymbol{c} \neq \boldsymbol{0}$, then its norm is separated from 0. This lemma is important in our proofs of hardness results for structured linear systems and packing LPs. We will prove it in Appendix A.2.

Lemma 2.2.5. *Let $\boldsymbol{A}$ and $\boldsymbol{c}$ a matrix and a vector such that $\|\boldsymbol{A}\boldsymbol{c}\|_2^2$ is integral. If $\boldsymbol{A}^\top \boldsymbol{c} \neq \boldsymbol{0}$, then*

$$\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}\|_2^2 \geq \frac{1}{\sigma_{\max}^2(\boldsymbol{A})} = \frac{1}{\lambda_{\max}\left(\boldsymbol{A}^\top \boldsymbol{A}\right)}.$$

Remark. We assume that in Problem (2.2) $\boldsymbol{A}$ in general has dimension $\widetilde{O}(n) \times n$, otherwise we can use subspace embedding [63] to compute an $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{\widetilde{O}(n)\times n}$ which approximately preserves the ℓ_2 norm of $\boldsymbol{A}\boldsymbol{x}$ for any $\boldsymbol{x}$. In addition, computing $\widetilde{\boldsymbol{A}}$ takes $\mathrm{nnz}(\boldsymbol{A})$ time, and it preserves the sparsity of $\boldsymbol{A}$.

2.3 Graph-Structured Block Matrices

In this subsection, we formally define Graph-Structured Block Matrices, and some matrices belonging to this class: 2-commodity Laplacian matrices, truss stiffness matrices, and 2 total variation matrices.

Suppose we have a collection of n disjoint sets of variables $X_1, \ldots, X_n$, with each set having the same size, $|X_i| = d$. Let $\boldsymbol{x}^i$ denote the vector of variables in X_i, and consider an equation of the form $\boldsymbol{M}\boldsymbol{x}^i - \boldsymbol{N}\boldsymbol{x}^j = 0$, where $\boldsymbol{M}$ and $\boldsymbol{N}$ are both $r \times d$ matrices. Now we form a linear system $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{0}$ by stacking m equations of the form given above as the rows of the system. Note that, very importantly, we allow a different choice of $\boldsymbol{M}$ and $\boldsymbol{N}$ for every pair of i and j. This matrix $\boldsymbol{B} \in \mathbb{R}^{mr\times nd}$ we refer to as an *Incidence-Structured Block Matrix* (ISBM), while we refer to $\boldsymbol{B}^\top \boldsymbol{B}$ as a *Graph-Structured Block Matrix* (GSBM). Note that $\boldsymbol{B}$ is not usually PSD, but $\boldsymbol{B}^\top \boldsymbol{B}$ is. The number of non-zeros in $\boldsymbol{B}^\top \boldsymbol{B}$ is $O(md^2)$. GSBMs come up in many applications, where we typically want to solve a linear system in the normal equations of $\boldsymbol{B}$.

Laplacian matrices are GSBMs where $d = 1$ and $\boldsymbol{M} = \boldsymbol{N} = w$, where w is a real number, and we allow different w for each pair of i and j. The corresponding ISBM for Laplacians is called an edge-vertex incidence matrix. Connection Laplacians are GSBMs where $d = O(1)$ and $\boldsymbol{M} = \boldsymbol{N}^\top = w\boldsymbol{Q}$, for some rotation matrix $\boldsymbol{Q}$ and a real num-

ber w. Again, we allow a different rotation matrix and scaling for every edge. For both Laplacians and Connection Laplacians, there exist linear system solvers that run in time $O(m \operatorname{polylog}(n, \epsilon^{-1}))$ and produce ϵ approximate solutions to the corresponding normal equations.

We now introduce several classes of ISBMs and their associated GSBMs.

Definition 2.3.1 (2-commodity incidence matrix)**.** *A* 2-commodity incidence matrix *is an ISBM where $d = 2$ and $r = 1$, and $\boldsymbol{M} = \boldsymbol{N}$, and we allow three types of $\boldsymbol{M}$: $\boldsymbol{M} = w \begin{pmatrix} 1 & 0 \end{pmatrix}$, $\boldsymbol{M} = w \begin{pmatrix} 0 & 1 \end{pmatrix}$ and $\boldsymbol{M} = w \begin{pmatrix} 1 & -1 \end{pmatrix}$, where in each case w is a real number which may depend on the pair i and j. We denote the set of all 2-commodity incidence matrices by $\mathcal{MC}_2$. The corresponding GSBM is called a 2-commodity Laplacian. The ISBM definition is equivalent to requiring the GSBM to have the form*

$$\boldsymbol{L}^1 \otimes \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \boldsymbol{L}^2 \otimes \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} + \boldsymbol{L}^{1+2} \otimes \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

where $\otimes$ is the tensor product and $\boldsymbol{L}^1$, $\boldsymbol{L}^2$, and $\boldsymbol{L}^{1+2}$ are all Laplacian matrices.

We adopt a convention that that the first variable in a set X_i is labelled $\boldsymbol{u}_i$ and the second is labelled $\boldsymbol{v}_i$. Using this convention, given a 2-commodity incidence matrix $\boldsymbol{B}$, the equation $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{0}$ must consist of scalings of the following three types of equations: $\boldsymbol{u}_i - \boldsymbol{u}_j = 0$, $\boldsymbol{v}_i - \boldsymbol{v}_j = 0$, and $\boldsymbol{u}_i - \boldsymbol{v}_i - (\boldsymbol{u}_j - \boldsymbol{v}_j) = 0$.

Definition 2.3.2 (Strict 2-commodity incidence matrix)**.** *A* strict 2-commodity incidence matrix *is a 2-commodity incidence matrix where the corresponding 2-commodity Laplacian has the property that $\boldsymbol{L}^1$, $\boldsymbol{L}^2$, and $\boldsymbol{L}^{1+2}$ all have the same non-zero pattern. We denote the set of all strict 2-commodity incidence matrices by $\mathcal{MC}_2^{>0}$. We denote the set of all strict 2-commodity incidence matrices with* integer entries *by $\mathcal{MC}_{2,\mathbb{Z}}^{>0}$.*

Linear systems in $\mathcal{MC}_2^{>0}$ are exactly the systems that one has to solve to when solving 2-commodity problems using Interior Point Methods (IPMs). For readers unfamiliar with

2-commodity problems or IPMs, we provide a brief explanation of why this is the case in Section 3.8. The $\mathcal{MC}_2^{>0}$ is more restrictive than $\mathcal{MC}_2$, and $\mathcal{MC}_{2,\mathbb{Z}}^{>0}$ in turn is even more restrictive. One could hope that fast linear system solvers exist for $\mathcal{MC}_2^{>0}$ or $\mathcal{MC}_{2,\mathbb{Z}}^{>0}$, even if they do not exist for $\mathcal{MC}_2$. However, our reductions show that even getting a fast approximate solver for $\mathcal{MC}_{2,\mathbb{Z}}^{>0}$ with polynomially bounded entries and condition number will lead to a fast solver for all matrices with polynomially bounded entries and condition number.

The next class we consider is Truss Stiffness Matrices. They have been studied extensively in the numerical linear algebra community [43, 64].

Definition 2.3.3 (2D Truss Incidence Matrices)**.** *Let $G = (V, E)$ be a graph whose vertices are n points in 2-dimension: $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^n \in \mathbb{R}^2$. Consider $X_1, \ldots, X_n$ where $d = 2$. A* 2D Truss Incidence Matrix *is an ISBM where $d = 2$ and $r = 1$, and for each i and j, $\boldsymbol{M} = w(\boldsymbol{s}^i - \boldsymbol{s}^j)^\top$, and w is a real number that may depend on the pair i and j, but $\boldsymbol{s}^i$ depends only on i and vice versa for $\boldsymbol{s}^j$. We denote the class of all 2D Truss Incidence Matrices by $\mathcal{T}_2$.*

In Section 2.4, we extend the above definition for 2D trusses into 3D.

Another important class of matrices is Total Variation Matrices (TV matrices). TV matrices come from Interior Point Methods for solving total variation minimization problem in image, see for example [65] and [32]. Not all TV matrices are GSBMs, but many GSBMs can be expressed as TV matrices.

Definition 2.3.4 (TV matrix and 2-TV Incidence Matrices)**.** *Let $E_1 \cup \ldots \cup E_s$ be a partition of the edge set of a graph. For each $1 \leq i \leq s$, let $\boldsymbol{B}^i$ be the edge-vertex incidence matrix of E_i, $\boldsymbol{W}^i$ be a diagonal matrix of edge weights, and $\boldsymbol{r}^i$ be a vector satisfying $\boldsymbol{W}^i \succcurlyeq \boldsymbol{r}^i(\boldsymbol{r}^i)^\top$. The* total variation matrix *(TV matrix) is defined as*

$$\boldsymbol{M} = \sum_{1 \leq i \leq s} (\boldsymbol{B}^i)^\top \left(\boldsymbol{W}^i - \boldsymbol{r}^i(\boldsymbol{r}^i)^\top \right) \boldsymbol{B}^i.$$

A 2-TV Incidence Matrix *is defined as any ISBM whose corresponding GSBM is a TV*

matrix with $\boldsymbol{W}^i \in \mathbb{R}^{2\times 2}$ *and* $\boldsymbol{r}^i \in \mathbb{R}^2$*. We denote the class of all 2-TV incidence matrices by* $\mathcal{V}_2$*.*

We remark that, given an Incidence-Structured Block Matrix $\boldsymbol{B} \in \mathbb{R}^{mr\times nd}$ with r, d being constant (this is usually the case), we can compute its Graph-Structured Block Matrix $\boldsymbol{B}^\top\boldsymbol{B}$ in $\mathrm{nnz}(\boldsymbol{B})$ time. Conversely, given a GSBM $\boldsymbol{M}$, we can decompose it to $\boldsymbol{M} = \boldsymbol{B}^\top\boldsymbol{B}$ for an ISBM $\boldsymbol{B}$ in $\mathrm{nnz}(\boldsymbol{M})$ time. Thus, transforming between ISBMs and GSBMs is never a bottleneck of running time.

2.4 Geometric Structures for 3-D Trusses

In this section, we give more details about the class of 3-D trusses which we will discuss in Section 5.

Tetrahedral meshes. For a subset $S \subset \mathbb{R}^3$, we define the *diameter* of S to be the maximum Euclidean distance between any pair of points in S. We define the *aspect ratio* of S to be the ratio between the radius of the smallest ball containing S and the radius of the largest ball inscribed in S. In mesh generation, aspect ratio is a common criterion for individual elements.

A tetrahedron is the convex hull of four non-coplanar points in $\mathbb{R}^3$. We will specify tetrahedrons in terms of sets of four such points. We refer to a set of tetrahedrons as a tetrahedral mesh.

We say a tetrahedral mesh is a *simplicial complex* if the intersection of every two tetrahedrons is either empty or a face of both two tetrahedrons. A simplicial complex is *convex* if the union of the images of its simplices is convex, as defined in [66].

The following definition characterizes rigidity and stiffness of a tetrahedral mesh, which is an adaption of Definition 2.3 in [44].

Definition 2.4.1. *The* rigidity graph *of a tetrahedral mesh is a graph whose vertices correspond to tetrahedrons and whose edges connect any two tetrahedrons sharing a triangle*

face. A tetrahedral mesh is stiffly-connected *iff (1) its rigidity graph is connected, and (2) for any vertex in the mesh, the rigidity subgraph induced on the tetrahedrons containing this vertex is connected.*

We define a *bounding box* of a convex 3D shape to be a 3D box such that: (1) the box contains all points of this shape, and (2) the volume of the box is same as the volume of this shape up to a constant factor. [67] gives a linear time algorithm which computes a bounding box of a convex shape in 3 dimensions.

Lemma 2.4.2. *(Lemma 3.6 of [67]) Given a 3D convex shape, one can compute in linear time a bounding box of this shape.*[1] *Moreover, the aspect ratio of the bounding box is same as the aspect ratio of the given shape up to a constant factor.*

3D trusses and stiffness matrices. We use the following notations for 3D trusses.

Definition 2.4.3 (3-dimensional truss)**.** 3-dimensional truss $\mathcal{T} = \langle V, \{\boldsymbol{p}_i\}_{i \in V}, T, E, \gamma \rangle$ *is given by*

- *A set of n vertices V embedded at distinct points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n \in \mathbb{R}^3$.*
- *A mesh (i.e. set) of tetrahedrons $T = \{t_1, t_2, \ldots\}$, each specified in terms of four vertices, i.e. we identify tetrahedron t_i with both four vertices $\{a_i, b_i, c_i, d_i\} \subseteq V$ and the convex hull of $\boldsymbol{p}_{a_i}, \boldsymbol{p}_{b_i}, \boldsymbol{p}_{c_i}, \boldsymbol{p}_{d_i}$.*
- *A set of edges E which is exactly the set of pairs of vertices that appear in some tetrahedron together. Each edge $e = (i, j) \in E$ represents a straight idealized bar between vertex points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$.*
- *A function $\gamma : E \to \mathbb{R}_+$, which assigns a stiffness coefficient $\gamma(e)$ to each edge e. The stiffness coefficient represents the stiffness of the idealized bar corresponding to edge e.*

[1] The lemma statement in [67] gives a bound related to the minimum-volume bounding box B^* of the input shape, but their proof uses the volume of the shape as a lower bound of the volume of B^*.

The following definition is a restatement and generalization of Definition 2.3.3. Definition 2.4.4 is more useful in our algorithm design in Section 5.

Definition 2.4.4 (3-D Truss stiffness matrix)**.** *Let $\mathcal{T} = \langle V, \{\boldsymbol{p}_i\}_{i\in V}, T, E, \gamma\rangle$ be a 3-dimensional truss. For each edge $e = (i, j) \in E$, we define an edge vector $\boldsymbol{b}^{(e)} \in \mathbb{R}^{3n}$ with 6 nonzero entries:*

$$\boldsymbol{b}^{(e)}_{3i-2:3i} = -\boldsymbol{b}^{(e)}_{3j-2:3j} = \frac{\boldsymbol{p}_i - \boldsymbol{p}_j}{\left\|\boldsymbol{p}_i - \boldsymbol{p}_j\right\|_2}.$$

The stiffness matrix *of the truss $\mathcal{T}$ is defined as*

$$\boldsymbol{A}_{\mathcal{T}} \stackrel{\text{def}}{=} \sum_{e=(i,j)\in E} \frac{\gamma(e)}{\left\|\boldsymbol{p}_i - \boldsymbol{p}_j\right\|_2} \boldsymbol{b}^{(e)} \boldsymbol{b}^{(e)\top}.$$

In addition, we use some terms to simplify our descriptions.

Definition 2.4.5. *A 3D truss is said to be* simple *iff it is a simplicial complex and every tetrahedron has bounded aspect ratio. We say a 3D truss is* edge-simple *iff its tetrahedral mesh is simple and every tetrahedron has bounded edge lengths and stiffness coefficients (i.e. both are bounded above and below by constants). We say a 3D truss is* convex edge-simple *iff it is edge-simple, and its tetrahedral mesh is convex.*

2.5 Techniques for Solving Linear Systems

Our algorithm combines two of the most important tools for solving linear systems: Nested dissection and preconditioning. Below, we give a brief introduction to some of the central results on these techniques.

Classic results due to Lipton, Rose, and Tarjan [48], and Miller and Thurston [49] combine to show that linear systems arising from simple tetrahedral meshes (see Definition 2.4.5) can be solved in $O(n^2)$ time. These results concern linear equations in an $n \times n$ matrix $\boldsymbol{A}$ where the indices $\{1, \ldots, n\}$ can be embedded as points $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ that form

the vertices of an explicitly given, simple tetrahedral mesh, and $\boldsymbol{A}_{ij}$ is non-zero only if the vertices i and j share an edge in the tetrahedral mesh.

Theorem 2.5.1 (Nested dissection [49]). *Let $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ be a symmetric matrix defined on a simple tetrahedral mesh. A Cholesky factorization $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{L}\boldsymbol{L}^\top \boldsymbol{P}^\top$ can be computed in time $O(n^2)$, in which $\boldsymbol{P}$ is a permutation matrix and $\boldsymbol{L}$ is a lower triangular matrix with $O(n^{4/3})$ nonzero entries. As a result, a linear system in $\boldsymbol{A}$ can be solved in time $O(n^2)$ by Gaussian elimination.*

Theorem 2.5.1 can be extended to a block matrix $\boldsymbol{A} \in \mathbb{R}^{cn \times cn}$ where c is a constant positive integer. Each vertex of the underlying graph corresponds to c indices of $\boldsymbol{A}$. In addition, the block corresponding to the column indices for vertex i and the row indices for j should be non-zero only if the vertices i and j share an edge in the tetrahedral mesh, or if $i = j$, i.e. when the block is on the diagonal.

The Nested Dissection algorithm relies on invoking *separators* recursively. A separator is a set of indices S such that the remaining indices $[n] \setminus S$ can be partitioned into two sets B and C such that every entry with $i \in B$ and $j \in C$ has $\boldsymbol{A}_{ij} = 0$. Furthermore, we guarantee that the partition is roughly balanced, for example, each of B and C contains no more than $\frac{3}{4} \cdot n$ indices. Nested Dissection recursively repeats the partitioning process on the union of each subset and the separator itself, that is, $B \cup S$ and $C \cup S$. Given such a recursive partition scheme, we reorder the indices of the matrix so that the indices in the separator S are eliminated *last*, and we then order the indices in B and C recursively in a similar way. We perform Gaussian elimination on the matrix according to this ordering, which only introduces a small fill-in size and few multiplication counts. This approach also works for eliminating a subset of the variables, resulting in a Schur complement on the rest.

Both the running time and representation cost of nested dissection algorithms are bottlenecked by the costs of the top-level separators. In Algorithm 12 TRUSSSOLVER, we will utilize improved running time bounds for nested dissection when better separators exist. The following lemma characterizes the performance of Nested Dissection given better

top-level separators.

Lemma 2.5.2. *Suppose we have a recursive separator decomposition of a simplicial complex with n bounded aspect ratio tetrahedrons such that:*

1. *the number of leaves, and hence total number of recursive calls, is at most n^{α}.*
2. *each leaf (bottom layer partition) has at most n^{β} tetrahedrons.*
3. *each top separator has size at most n^{γ}.*

Then we can find an exact Cholesky factorization of the associated stiffness matrix in time $O(n^{\alpha+2\beta} + n^{\alpha+3\gamma})$, and the total resulting fill-in is $O(n^{\alpha+\frac{4}{3}\beta} + n^{\alpha+2\gamma})$.

We give a proof of the above lemma in Section 5.5.1, which is an adaption of the analysis in [48] We remark that the algorithmic realization of this can be viewed as utilizing the nested dissection algorithm 2.5.1 to complete this structure into a full separator tree.

Last but not least, we state the following theorem for preconditioned conjugate gradient, which will be used in bounding the running time of our algorithm.

Theorem 2.5.3 (Preconditioned conjugate gradient [68])**.** *Let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$ be two symmetric positive semidefinite matrices and let $\boldsymbol{b} \in \mathbb{R}^{n}$. Each iteration of the preconditioned conjugate gradient multiplies one vector by $\boldsymbol{A}$, solves one linear system in $\boldsymbol{B}$, and performs a constant number of vector additions. For any $\epsilon > 0$, the algorithm outputs an $\boldsymbol{x}$ satisfying $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \leq \epsilon \|\boldsymbol{b}\|_2$ in $O(\sqrt{\kappa(\boldsymbol{A}, \boldsymbol{B})} \log(1/\epsilon))$ such iterations.*

We remark that while there are settings where the convergence of preconditioned conjugate gradient is numerically unstable, the eigenvalue-based bound that we utilize here is stable once the solves involving $\boldsymbol{B}$ have polynomially small errors.

CHAPTER 3

HARDNESS RESULTS OF STRUCTURED LINEAR SYSTEMS

In this chapter, we prove Theorem 1.1.1.

3.1 Measuring the difficulty of solving a linear system

Running times for iterative linear system solvers generally depend on the number of non-zeros in the input matrix, the condition number of the input matrix, the accuracy, and the bit complexity.

In this section, we formally define several measures of complexity of the linear systems we use. This is crucial, because we want to make sure that our reductions do not rely on mapping into extremely ill-conditioned matrices, and so we use these measures to show that this is in fact not the case.

Definition 3.1.1. *The sparse parameter complexity of an* LSA *instance* $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$ *where* $\boldsymbol{A} \in \mathbb{Z}^{m \times n}$ *and* $\operatorname{nnz}(\boldsymbol{A}) \geq \max(m, n)$*, and* $\epsilon > 0$*, is*

$$\mathcal{S}(\boldsymbol{A}, \boldsymbol{c}, \epsilon) \stackrel{\text{def}}{=} \left(\operatorname{nnz}(\boldsymbol{A}), \max\left(\|\boldsymbol{A}\|_{\max}, \|\boldsymbol{c}\|_{\max}, \frac{1}{\min_+(\boldsymbol{A})}, \frac{1}{\min_+(\boldsymbol{c})} \right), \kappa(\boldsymbol{A}), \epsilon^{-1} \right).$$

Note in the definition above that when $\boldsymbol{A} \neq \boldsymbol{0}$ and $\boldsymbol{c} \neq \boldsymbol{0}$ have only integer entries, we trivially have $\min_+(\boldsymbol{A}) \geq 1$ and $\min_+(\boldsymbol{c}) \geq 1$. However, including $\frac{1}{\min_+(\boldsymbol{A})}$, and $\frac{1}{\min_+(\boldsymbol{c})}$ in the definition stated above is useful when working with intermediate matrices whose entries are not integer valued.

The following Claim bounds some frequently used quantities by parameters defined in Definition 3.1.1. We prove it in Appendix A.2.

Claim 3.1.2. *Given an* $\text{LSA}(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$ *satisfying:* $\boldsymbol{A}$ *and* $\boldsymbol{c}$ *are integral,* $\boldsymbol{A}$ *has* s *non-zeros and condition number* K*, and every entry of* $\boldsymbol{A}$ *and* $\boldsymbol{c}$ *has absolute value at most* U*. Then*

1. $\|\boldsymbol{A}\|_\infty \leq sU$.

2. $\|\boldsymbol{c}\|_2 \leq \sqrt{s}U$.

3. $\frac{1}{sU^2} \leq \lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A}) \leq sU^2$ *and* $\lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A}) \geq \frac{1}{sK^2U^2}$.

We use the term *matrix class* to refer to an infinite set of matrices $\mathcal{M}$. In this section, we formally define a notion of efficient reduction between linear systems in different classes of matrices.

Definition 3.1.3 (Efficient f-reducibility)**.** *Suppose we have two matrix classes $\mathcal{M}_1$ and $\mathcal{M}_2$, and there exist two algorithms $\mathcal{A}_{1\to 2}$ and $\mathcal{A}_{1\leftarrow 2}$ s.t. given an* LSA *instance $(\boldsymbol{M}^1, \boldsymbol{c}^1, \epsilon)$, where $\boldsymbol{M}^1 \in \mathcal{M}_1$, the call $\mathcal{A}_{1\to 2}(\boldsymbol{M}^1, \boldsymbol{c}^1, \epsilon_1)$ returns an* LSA *instance $(\boldsymbol{M}^2, \boldsymbol{c}^2, \epsilon_2)$ s.t. if $\boldsymbol{x}^2$ is a solution to* LSA *instance $(\boldsymbol{M}^2, \boldsymbol{c}^2, \epsilon_2)$ then $\boldsymbol{x}^1 = \mathcal{A}_{1\leftarrow 2}(\boldsymbol{M}^1, \boldsymbol{M}^2, \boldsymbol{x}^2)$ is a solution to* LSA *instance $(\boldsymbol{M}^1, \boldsymbol{c}^1, \epsilon_1)$.*

Consider a function of $f : \mathbb{R}_+^4 \to \mathbb{R}_+^4$ s.t. every output coordinate is an increasing function of every input coordinate. Suppose that we always have

$$\mathcal{S}(\boldsymbol{M}^2, \boldsymbol{c}^2, \epsilon_2) \leq f(\mathcal{S}(\boldsymbol{M}^1, \boldsymbol{c}^1, \epsilon_1)),$$

and the running times of $\mathcal{A}_{1\to 2}(\boldsymbol{M}^1, \boldsymbol{c}^1, \epsilon_1)$ and $\mathcal{A}_{1\leftarrow 2}(\boldsymbol{M}^1, \boldsymbol{M}^2, \boldsymbol{x}^2)$ are both bounded by $O(\mathrm{nnz}(\boldsymbol{M}^1))$.

Then we say that $\mathcal{M}_1$ is efficiently f-reducible to $\mathcal{M}_2$, which we also write as

$$\mathcal{M}_1 \leq_f \mathcal{M}_2.$$

Lemma 3.1.4. *Suppose $\mathcal{M}_1 \leq_f \mathcal{M}_2$ and $\mathcal{M}_2 \leq_g \mathcal{M}_3$. Then $\mathcal{M}_1 \leq_{g\circ f} \mathcal{M}_3$.*

Proof. The proof is simply by the trivial composition of the two reductions. □

Definition 3.1.5. *We let $\mathcal{G}$ denote the class of all matrices with integer valued entries s.t. there is at least one non-zero entry in every row and column*[1].

3.2 Main Results and Reduction Outline

In this section, we use the notions of sparse parameter complexity and matrix class reductions defined in Section 3.1 to prove our main technical result, Theorem 3.2.1. Then, as a corollary, we get Theorem 3.2.2.

Theorem 3.2.1. *Let $f(s, U, K, \epsilon) = (O(s \log(sU)), \text{poly}(UK\epsilon^{-1}s), \text{poly}(UK\epsilon^{-1}s), \text{poly}(UK\epsilon^{-1}s))$, then*

1. $\mathcal{G} \leq_f \mathcal{MC}^{>0}_{2,\mathbb{Z}}$.
2. $\mathcal{G} \leq_f \mathcal{T}_2$.
3. $\mathcal{G} \leq_f \mathcal{V}_2$.

Theorem 3.2.2. *Suppose we have an algorithm which solves every Linear System Approximation Problem $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$ with sparse parameter complexity $\mathcal{S}(\boldsymbol{A}, \boldsymbol{c}, \epsilon) \leq (s, U, K, \epsilon^{-1})$ $O(s^a \,\text{polylog}(s, U, K, \epsilon^{-1}))$ for some $a \geq 1$, whenever $\boldsymbol{A} \in \mathcal{R}$ for at least one of $\mathcal{R} \in \left\{\mathcal{MC}^{>0}_{2,\mathbb{Z}}, \mathcal{T}_2, \mathcal{V}_2\right\}$. I.e. we have a "fast" solver*[2] *for one of the matrix classes $\mathcal{MC}^{>0}_{2,\mathbb{Z}}, \mathcal{T}_2$, or $\mathcal{V}_2$. Then every Linear System Approximation Problem $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$ where $\boldsymbol{A} \in \mathcal{G}$ with sparse parameter complexity $\mathcal{S}(\boldsymbol{A}, \boldsymbol{c}, \epsilon) \leq (s, U, K, \epsilon^{-1})$ can be solved in time $O(s^a \,\text{polylog}(s, U, K, \epsilon^{-1}))$.*

Proof. The theorem is a immediate corollary of Theorem 3.2.1. □

Definition 3.2.3. *We let $\mathcal{G}_{z,2}$ denote the class of all matrices with integer valued entries s.t. there is at least one non-zero entry in every row and column, and every row has zero row sum, and for each row, the sum of the positive coefficients is a power of 2.*

[1] If there is a row or column with only zeros, then it can always be handled trivially in the context of solving linear systems

[2] The reduction requires only a single linear system solve, and uses the solution in a black-box way. So the reduction also applies if the solver for the class $\mathcal{R}$ only works with high probability or only has running time guarantees in expectation.

Our reduction follows the steps:

$$\mathcal{G} \to \mathcal{G}_{z,2} \to \mathcal{MC}_2 \to \mathcal{MC}_2^{>0} \to \mathcal{MC}_{2,\mathbb{Z}}^{>0}.$$

Recall that, from Definition 2.3.2, $\mathcal{MC}_2^{>0}$ is the class of strict 2-commodity matrices, and $\mathcal{MC}_{2,\mathbb{Z}}^{>0}$ is the class of strict 2-commodity matrices with integral entries. Specifically, we will prove the following sequence of lemmas.

Lemma 3.2.4. *Let* $f(s,U,K,\epsilon) = \left(O(s), O\left(\epsilon^{-1}s^{9/2}U^3\right), O\left(\epsilon^{-1}s^8U^3K\right), O\left(s^{5/2}U^2\epsilon^{-1}\right)\right)$, *then*

$$\mathcal{G} \leq_f \mathcal{G}_{z,2}.$$

Lemma 3.2.5. *Let* $f(s,U,K,\epsilon) = \Big(O(s\log(sU)), O\left(s^{3/2}U\log^{1/2}(sU)\right),$
$O\left(Ks^4U^2\log^2(sU)\right), O\left(sU^2\epsilon^{-1}\right)\Big)$, *then*

$$\mathcal{G}_{z,2} \leq_f \mathcal{MC}_2.$$

Lemma 3.2.6. *Let* $f(s,U,K,\epsilon) = (O(s), O\left(\epsilon^{-1}U^2K\right), O\left(\epsilon^{-1}s^2U^2K\right), O(\epsilon^{-1}))$, *then*

$$\mathcal{MC}_2 \leq_f \mathcal{MC}_2^{>0}.$$

Lemma 3.2.7. *Let* $f(s,U,K,\epsilon) = (s, \epsilon^{-1}sU, 2K, O(\epsilon^{-1}))$, *then*

$$\mathcal{MC}_2^{>0} \leq_f \mathcal{MC}_{2,\mathbb{Z}}^{>0}.$$

Lemma 3.2.8. *Let* $f(s,U,K,\epsilon)$ *be as defined in Lemma 3.2.5 then*

$$\mathcal{G}_{z,2} \leq_f \mathcal{T}_2.$$

Lemma 3.2.9. *Let* $f(s, U, K, \epsilon) = (s, U, K, \epsilon^{-1})$*, then*

$$\mathcal{MC}_2 \leq_f \mathcal{V}_2.$$

Proof of Theorem 3.2.1. Follows by appropriate composition (Lemma 3.1.4) applied to the the Lemmas above, i.e. 3.2.4, 3.2.5, 3.2.6, 3.2.7, 3.2.8 and 3.2.9. □

Outline of the Remaining Section. We prove Lemma 3.2.4 in Section 3.6, Lemma 3.2.5 in Section 3.3, Lemma 3.2.6 in Section 3.4, Lemma 3.2.7 in Section 3.5, Lemma 3.2.8 in Section 3.7, and Lemma 3.2.9 in Section 3.8.2. In addition, we show how to derive multi-commodity Laplacians and Total Variation matrices by Interior-point method in Section 3.8.

3.3 Reducing Zero-Sum Power Two Linear Systems to Two-Commodity Linear Systems

To prove Lemma 3.2.5, we need to provide mapping algorithms $\mathcal{A}_{\mathcal{G}_{\text{z},2} \to \mathcal{MC}_2}$ for mapping linear system approximation (LSA) instances over matrices in $\mathcal{G}_{\text{z},2}$ to LSA problem instances over matrices in $\mathcal{MC}_2$, as well as $\mathcal{A}_{\mathcal{MC}_2 \leftarrow \mathcal{G}_{\text{z},2}}$ for mapping the resulting solutions back. These leads to the following main components:

- Algorithm 1 states the pseudo-code for the algorithm REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{MC}_2$, which implements the desired mapping of problem instances. Given LSA problem instance $(\boldsymbol{A}, \boldsymbol{c}^{\text{A}}, \epsilon^{\text{A}})$ where $\boldsymbol{A} \in \mathcal{G}_{\text{z},2}$, the call REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{MC}_2(\boldsymbol{A}, \boldsymbol{c}^{\text{A}}, \epsilon^{\text{A}})$ returns an LSA problem instance $(\boldsymbol{B}, \boldsymbol{c}^{B}, \epsilon^{\text{B}})$ where $\boldsymbol{B} \in \mathcal{MC}_2$. (Strictly speaking, REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{MC}_2$ also has a parameter α which we will set before using the algorithm.)

- Algorithm 2 provides the pseudo-code for $\mathcal{MC}_2$Gadget, a short subroutine used in REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{MC}_2$to represent the equation $a + b = 2c$ using two-commodity

constraints.

- Algorithm 3 provides the (trivial) pseudo-code for MAPSOLN $\mathcal{MC}_2$TO$\mathcal{G}_{z,2}$ used to map solutions to LSA problems over $\mathcal{MC}_2$ back to solutions over $\mathcal{G}_{z,2}$ by restricting onto the original variables.

Pseducode of the key reduction routine that creates the new linear system, REDUCE $\mathcal{G}_{z,2}$TO$\mathcal{MC}_2$, is shown in Algorithm 1. Note that given two singleton multi-sets each containing a single equation, e.g. $\left\{\boldsymbol{a}_1^\top \boldsymbol{x} = c_1\right\}$ and $\left\{\boldsymbol{a}_2^\top \boldsymbol{x} = c_2\right\}$ where $\boldsymbol{a}_1, \boldsymbol{a}_2$ are vectors, we define $\left\{\boldsymbol{a}_1^\top \boldsymbol{x} = c_1\right\} + \left\{\boldsymbol{a}_2^\top \boldsymbol{x} = c_2\right\} = \left\{\boldsymbol{a}_1^\top \boldsymbol{x} + \boldsymbol{a}_2^\top \boldsymbol{x} = c_1 + c_2\right\}$ and we define $\left\{\boldsymbol{a}_1^\top \boldsymbol{x} = c_1\right\} \cup \left\{\boldsymbol{a}_2^\top \boldsymbol{x} = c_2\right\} = \left\{\boldsymbol{a}_1^\top \boldsymbol{x} = c_1, \boldsymbol{a}_2^\top \boldsymbol{x} = c_2\right\}$.

Algorithm 1 REDUCE $\mathcal{G}_{\text{z},2}$ TO $\mathcal{MC}_2$

Input: $(\boldsymbol{A}, \boldsymbol{c}^{\text{A}}, \epsilon_1)$ where $\boldsymbol{A} \in \mathcal{G}_{\text{z},2}$ is an $m \times n$ matrix, $\boldsymbol{c}^{\text{A}} \in \mathbb{R}^n$, $0 < \epsilon_1 < 1$, and $\alpha > 0$.
Output: $(\boldsymbol{B}, \boldsymbol{c}^{\text{B}}, \epsilon_2)$ where $\boldsymbol{B} \in \mathcal{MC}_2$ is an $m \times n$ matrix, $\boldsymbol{c}^{\text{B}} \in \mathbb{R}^n$, and $0 < \epsilon_2 < 1$.

$\epsilon_2 \leftarrow \epsilon_1 \left(1 + \frac{1}{\alpha}\right)^{-1/2} \left(1 + \frac{\left\|\boldsymbol{c}^{\text{A}}\right\|_2^2 \sigma_{\max}^2(\boldsymbol{A})}{\alpha+1}\right)^{-1/2}$
$\mathcal{X} \leftarrow \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ {$\mathcal{MC}_2$ variables and index of new variables}
Let $\boldsymbol{x}$ be the vector of variables corresponding to the set of variables $\mathcal{X}$
$t \leftarrow n + 1$
Initialize $\mathcal{A} \leftarrow \emptyset$, $\mathcal{B} \leftarrow \emptyset$ {Multisets of main and $\mathcal{MC}_2$ auxiliary equations respectively}
for each equation $1 \le i \le m$ in $\boldsymbol{A}$ **do**
  Let $\mathcal{I}^{+1} \leftarrow \{j : \boldsymbol{A}_{ij} > 0\}, \mathcal{I}^{-1} \leftarrow \{j : \boldsymbol{A}_{ij} < 0\}$
  if $|\mathcal{I}^{+1}| = 1$ and $|\mathcal{I}^{-1}| = 1$ **then**
    Let the only elements in $\mathcal{I}^{+1}$ and $\mathcal{I}^{-1}$ be j_+ and j_- respectively. {Note $\boldsymbol{A}_{ij_+} = -\boldsymbol{A}_{ij_-}$}
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\boldsymbol{A}_{ij_+}\boldsymbol{u}_t - \boldsymbol{A}_{ij_+}\boldsymbol{u}_{j_-} = \boldsymbol{c}_i\}$
    $w_i \leftarrow \alpha$
    $\mathcal{B} \leftarrow \mathcal{B} \cup w_i^{1/2} \cdot \{\boldsymbol{A}_{ij_+}\boldsymbol{u}_{j_+} - \boldsymbol{A}_{ij_+}\boldsymbol{u}_t = 0\}$
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{\boldsymbol{u}_t, \boldsymbol{v}_t\}$, update $\boldsymbol{x}$ accordingly
    $t \leftarrow t + 1$
  else
    $\mathcal{A}_i \leftarrow \{\boldsymbol{A}_i\boldsymbol{u} = \boldsymbol{c}_i\}$
    $\mathcal{B}_i \leftarrow \emptyset$
    for $s = -1, +1$ **do**
      $r \leftarrow 0$
      while $\mathcal{A}_i$ has strictly more than 1 coefficient with sign s **do**
        For each j, let $\widehat{\boldsymbol{A}}_{ij}$ be the coefficient of $\boldsymbol{u}_j$ in $\mathcal{A}_i$.
        $\mathcal{I}^s_{odd} \leftarrow \{j \in \mathcal{I}^s : \lfloor|\widehat{\boldsymbol{A}}_{ij}|/2^r\rfloor \text{ is odd}\}$
        Pair the indices of $\mathcal{I}^s_{odd}$ into k disjoint pairs (j_k, l_k)
        for each pair of indices (j_k, l_k) **do**
          $\mathcal{A}_i \leftarrow \mathcal{A}_i + s \cdot 2^r \{(2\boldsymbol{u}_t - (\boldsymbol{u}_{j_k} + \boldsymbol{u}_{l_k})) = 0\}$
          $\mathcal{B}_i \leftarrow \mathcal{B}_i \cup -s \cdot 2^r \cdot \mathcal{MC}_2\text{GADGET}(\boldsymbol{u}_{j_k}, \boldsymbol{u}_{l_k}, t)$
          $\mathcal{X} \leftarrow \mathcal{X} \cup \{\boldsymbol{u}_t, \ldots, \boldsymbol{u}_{t+6}, \boldsymbol{v}_t, \ldots, \boldsymbol{v}_{t+6}\}$, update $\boldsymbol{x}$ accordingly
          $t \leftarrow t + 7$
          $r \leftarrow r + 1$
        end for
      end while
    end for
    $w_i \leftarrow \alpha\,|\mathcal{B}_i|$
    $\mathcal{B} \leftarrow \mathcal{B} \cup w_i^{1/2} \cdot \mathcal{B}_i$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}_i$.
  end if
end for
return ϵ_2 and $\boldsymbol{B}$, $\boldsymbol{c}$ s.t. $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{c}$ corresponds to the equations in $\mathcal{A} \cup \mathcal{B}$, on the variable set $\mathcal{X}$.

The central object created by Algorithm 1 is the matrix $\boldsymbol{B}$, which contains both new equations and new variables. We will superscript the variables with ^{A} to distinguish variables appear in th original equation $\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}} = \boldsymbol{c}^{\mathrm{A}}$ from new variables. We will term the new variables as $\boldsymbol{x}^{\text{aux}}$, and write a vector solution for the new problem, $\boldsymbol{x}^{\mathrm{B}}$, as:

$$\boldsymbol{x}^{\mathrm{B}} = \begin{pmatrix} \boldsymbol{x}^{\mathrm{A}} \\ \boldsymbol{x}^{\text{aux}} \end{pmatrix}. \tag{3.1}$$

Let n_A be the dimension of $\boldsymbol{x}^{\mathrm{A}}$, and n_B be the dimension of $\boldsymbol{x}^{\mathrm{B}}$, respectively. We order the variables so that for an appropriately chosen index h we have

1. $\boldsymbol{x}^{\text{aux}}_{1:h}$ corresponds to the $\boldsymbol{u}$-coordinates of the auxiliary variables created in $\mathcal{MC}_2$-gadgets.

2. $\boldsymbol{x}^{\text{aux}}_{h+1:n_B-n_A}$ corresponds to the $\boldsymbol{v}$-coordinates of the auxiliary variables created in $\mathcal{MC}_2$-gadgets.

With this ordering $\boldsymbol{x}^{\mathrm{B}}_t = \boldsymbol{u}_t$ for $t \leq n_A + h$.

Furthermore, we will distinguish the equations in $\boldsymbol{B}$ into ones formed from manipulating $\boldsymbol{A}$, i.e. the equations added to the set $\mathcal{A}$, from the auxiliary equations, i.e. the equations added to the set $\mathcal{B}$. We use $\boldsymbol{W}^{1/2} = \text{DIAG}_i(w_i^{1/2})$ to refer to the diagonal matrix of weights w_i applied to the auxiliary equations $\mathcal{B}$ in Algorithm 1. In Algorithm 1, a real value $\alpha > 0$ is set initially and used when computing the weights $w_i^{1/2}$. For convenience, thoughout most of this section, we will treat α as an arbitrary constant, and only eventually substitute in its value to complete our main proof.

This leads to the following representation of $\boldsymbol{B}$ and $\boldsymbol{c}^{\mathrm{B}}$ which we will use throughout our analysis of the algorithm:

$$\boldsymbol{B} = \begin{pmatrix} \widehat{\boldsymbol{A}} \\ \boldsymbol{W}^{1/2}\widehat{\boldsymbol{B}} \end{pmatrix}. \tag{3.2}$$

Here the equations of $\widehat{\boldsymbol{A}}$ corresponds to $\mathcal{A}$ in REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{MC}_2$, and $\widehat{\boldsymbol{B}}$ corresponds to the auxiliary constraints, i.e. equations of $\mathcal{B}$ in REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{MC}_2$. Also, the vector $\boldsymbol{c}^{\text{B}}$ created is simply an extension of $\boldsymbol{c}^{\text{A}}$:

$$\boldsymbol{c}^{\text{B}} = \begin{pmatrix} \boldsymbol{c}^{\text{A}} \\ \mathbf{0} \end{pmatrix}. \tag{3.3}$$

Finally, as Algorithm 1 creates new equations for each row of $\boldsymbol{A}$ independently, we will use S_i to denote the subset of indices of the rows of $\widehat{\boldsymbol{B}}$ that's created from $\boldsymbol{A}_{i,:}$ aka. the the auxiliary constraints generated from the call to $\mathcal{MC}_2$GADGET upon processing $\boldsymbol{A}_{i,:}$. We will also denote the size of these using

$$m_i \stackrel{\text{def}}{=} |S_i|,$$

and use $\widehat{\boldsymbol{B}}_i$ to denote these part of $\widehat{\boldsymbol{B}}$ that corresponds to these rows. The Gadget routine used in the reduction, and the (trivial) solution mapper are stated below.

Algorithm 2 $\mathcal{MC}_2$GADGET

Input: Scalar variables $\boldsymbol{u}_{j_k}, \boldsymbol{u}_{l_k}$, integer t.

Output: Set of equations for a multi-commodity gadget that computes the average of $\boldsymbol{u}_{j_k}$ and $\boldsymbol{u}_{l_k}$.

1: **return**

$$\begin{aligned}
\{\boldsymbol{u}_{t+3} - \boldsymbol{v}_{t+3} - (\boldsymbol{u}_{t+4} - \boldsymbol{v}_{t+4}) &= 0,\\
\boldsymbol{u}_t - \boldsymbol{u}_{t+3} &= 0,\\
\boldsymbol{u}_{t+4} - \boldsymbol{u}_{j_k} &= 0,\\
\boldsymbol{v}_{t+3} - \boldsymbol{v}_{t+1} &= 0,\\
\boldsymbol{v}_{t+2} - \boldsymbol{v}_{t+4} &= 0,\\
\boldsymbol{u}_{t+5} - \boldsymbol{v}_{t+5} - (\boldsymbol{u}_{t+6} - \boldsymbol{v}_{t+6}) &= 0,\\
\boldsymbol{u}_t - \boldsymbol{u}_{t+5} &= 0,\\
\boldsymbol{u}_{t+6} - \boldsymbol{u}_{l_k} &= 0,\\
\boldsymbol{v}_{t+5} - \boldsymbol{v}_{t+2} &= 0,\\
\boldsymbol{v}_{t+1} - \boldsymbol{v}_{t+6} &= 0\}
\end{aligned}$$

Algorithm 3 MAPSOLN $\mathcal{MC}_2$TO$\mathcal{G}_{\text{z},2}$

Input: $m \times n$ matrix $\boldsymbol{A} \in \mathcal{G}_{\text{z},2}$, $m' \times n'$ matrix $\boldsymbol{B} \in \mathcal{MC}_2$, vector $\boldsymbol{c}^{\text{A}} \in \mathbb{R}^m$, vector $\boldsymbol{x}^{\text{B}} \in \mathbb{R}^{n'}$.

Output: Vector $\boldsymbol{x}^{\text{A}} \in \mathbb{R}^n$.

1: **if** $\boldsymbol{A}^\top \boldsymbol{c}^{\text{A}} = \boldsymbol{0}$ **then**
2: **return** $\boldsymbol{x}^{\text{A}} \leftarrow \boldsymbol{0}$
3: **else**
4: **return** $\boldsymbol{x}^{\text{A}} \leftarrow \boldsymbol{x}^{\text{B}}_{1:n}$
5: **end if**

We upper bound the number of nonzero entries of $\boldsymbol{B}$ by the number of nonzero entries of $\boldsymbol{A}$.

Lemma 3.3.1. $\operatorname{nnz}(\boldsymbol{B}) = O\left(\operatorname{nnz}(\boldsymbol{A}) \log \|\boldsymbol{A}\|_\infty\right)$ *and both dimensions of* $\boldsymbol{B}$ *are* $O\left(\operatorname{nnz}(\boldsymbol{A}) \log \|\boldsymbol{A}\|_\infty\right)$.

Proof. Since Algorithm 1 constructs new equations for each row of $\boldsymbol{A}$ independently, we bound the number of new variables and new equations (that is, the size of the submatrix $\begin{pmatrix} \widehat{\boldsymbol{A}}_i \\ \widehat{\boldsymbol{B}}_i \end{pmatrix}$) for each row i of $\boldsymbol{A}$ separately.

Let n_i be the number of nonzero entries of the ith row of $\boldsymbol{A}$. We count the number of variables in the equation $\mathcal{A}_i$, at each iteration of the while-loop in line 20 of Algorithm 1. Let $X^{(r)}$ be the subset of variables with nonzero coefficients in $\mathcal{A}_i$, at the end of iteration r. Let $X^{(r)}_{aux}$ be the subset of $X^{(r)}$ containing all auxiliary variables created in iteration r.

Note in each iteration, Algorithm 1 replaces two variables by a new auxiliary variable. It gives that

$$\left|X^{(1)}_{aux}\right| \leq \frac{n_i}{2}, \text{ and } \left|X^{(1)} \setminus X^{(1)}_{aux}\right| \leq n_i.$$

Since each auxiliary variable in $X^{(1)}_{aux}$ has coefficient 2, in the 2nd iteration, together with another variable of coefficient 2, it must be replaced by a new auxiliary variable. Thus, at the end of the 2nd iteration, all auxiliary variables $X^{(1)}_{aux}$ will not appear in the equation $\mathcal{A}_i$. This implies that

$$\left|X^{(2)}_{aux}\right| \leq \frac{n_i}{2} + \frac{n_i}{4}, \text{ and } \left|X^{(2)} \setminus X^{(2)}_{aux}\right| \leq n_i.$$

Similarly, at the end of the t^{th} iteration, we have

$$\left|X_{aux}^{(r)}\right| \leq \sum_{1 \leq s \leq r} \frac{n_i}{2^s} \leq n_i,$$

and

$$\left|X^{(r)} \setminus X_{aux}^{(r)}\right| \leq n_i.$$

Since Algorithm 1 pulls out a factor 2 in each iteration, the total number of iterations is at most $\log \|\boldsymbol{A}_i\|_1$. Since each auxiliary variable in $\mathcal{A}_i$ during the construction corresponds to $O(1)$ auxiliary variables and $O(1)$ equations in $\widehat{\boldsymbol{B}}_i$, the total number of auxiliary variables and equations for row i of $\boldsymbol{A}$ is

$$O\left(n_i \log \|\boldsymbol{A}_i\|_1\right).$$

Therefore, the number of variables and the number of equations in $\boldsymbol{B}$ (that is, the both dimensions of $\boldsymbol{B}$) are

$$O\left(\operatorname{nnz}(\boldsymbol{A}) \log \|\boldsymbol{A}\|_\infty\right).$$

Since each row of $\boldsymbol{B}$ has $O(1)$ nonzero coefficients, we have

$$\operatorname{nnz}(\boldsymbol{B}) = O\left(\operatorname{nnz}(\boldsymbol{A}) \log \|\boldsymbol{A}\|_\infty\right).$$

This completes the proof. □

Reduction Between Exact Solvers

The most important relation between $\boldsymbol{A}$ and $\boldsymbol{B}$ is given by the following claim.

Claim 3.3.2 (Reduction between exact solvers)**.** *Fix any* $\boldsymbol{x}^A \in \mathbb{R}^n$*. Then*

$$\left\|\boldsymbol{A}\boldsymbol{x}^A - \boldsymbol{c}^A\right\|_2^2 = \frac{\alpha+1}{\alpha} \min_{\boldsymbol{x}^{aux}} \left\| \boldsymbol{B} \begin{pmatrix} \boldsymbol{x}^A \\ \boldsymbol{x}^{aux} \end{pmatrix} - \boldsymbol{c}^B \right\|_2^2 .$$

As a Corollary of Claim 3.3.2, we observe the following:

Lemma 3.3.3. *Given* LSA *problem instance* $(\boldsymbol{A}, \boldsymbol{c}^A, \epsilon^A)$ *where* $\boldsymbol{A} \in \mathcal{G}_{z,2}$*, suppose*

$$(\boldsymbol{B}, \boldsymbol{c}^B, \epsilon^B) = \textsc{Reduce}\, \mathcal{G}_{z,2}\textsc{to}\mathcal{MC}_2(\boldsymbol{A}, \boldsymbol{c}^A, \epsilon^A).$$

Then $\boldsymbol{B} \in \mathcal{MC}_2$ *and if* $\boldsymbol{x}^B = \begin{pmatrix} \boldsymbol{x}^A \\ \boldsymbol{x}^{aux} \end{pmatrix}$ *is a solution to the* exact LSA *problem* $(\boldsymbol{B}, \boldsymbol{c}^B, 0)$*, then* $\boldsymbol{x}^A$ *is a solution to the* exact LSA *problem* $(\boldsymbol{A}, \boldsymbol{c}^A, 0)$.

Proof. This follows immediately from minimizing over $\boldsymbol{x}^{\mathrm{A}}$ on both sides of the equation established by Claim 3.3.2 and then applying and Fact 2.2.2. □

Before proving Claim 3.3.2 we first note a basic guarantee obtained by $\textsc{Reduce}\, \mathcal{G}_{\mathrm{z},2}\textsc{to}\mathcal{MC}_2$:

$$\mathbf{1}^\top \left(\begin{pmatrix} \widehat{\boldsymbol{A}}_i \\ \widehat{\boldsymbol{B}}_i \end{pmatrix} \boldsymbol{x}^{\mathrm{B}} - \begin{pmatrix} \boldsymbol{c}_i^{\mathrm{A}} \\ \mathbf{0} \end{pmatrix} \right) = \boldsymbol{A}_i \boldsymbol{x}^{\mathrm{A}} - \boldsymbol{c}_i^{\mathrm{A}}. \tag{3.4}$$

To verify this guarantee, we consider two cases separately. The first case is when the condition $|\mathcal{I}^{+1}| = 1$ AND $|\mathcal{I}^{-1}| = 1$ is true (see Algorithm 1, Line 7). In this case, the main constraint in the output corresponding to row i is $\{\boldsymbol{A}_{ij_+}\boldsymbol{x}_t^{\mathrm{B}} - \boldsymbol{A}_{ij_+}\boldsymbol{x}_{j_-}^{\mathrm{B}} = \boldsymbol{c}_i^{\mathrm{A}}\}$, while the auxiliary constraints contain only a single row $\{\boldsymbol{A}_{ij_+}\boldsymbol{x}_{j_+}^{\mathrm{B}} - \boldsymbol{A}_{ij_+}\boldsymbol{x}_t^{\mathrm{B}} = 0\}$, and adding these proves the guarantee for this case. The second case is when the condition in Algorithm 1, Line 7) is false. We consider the case $s = +1$. The case $s = -1$ is proved similarly. Note that we will refer to variables j_k and l_k only in the context of a fixed value of t, which always ensures that they are unambiguously defined. In the case $s = +1$, each time

we modify $\mathcal{A}_i$ by adding $2^r\left(2\boldsymbol{x}^{\mathrm{B}}_{t+1} - \left(\boldsymbol{x}^{\mathrm{B}}_{j_k} + \boldsymbol{x}^{\mathrm{B}}_{l_k}\right)\right) = 0$, we we also use $\mathcal{MC}_2$Gadget to create auxiliary constraints that sum up to exactly $-2^r\left(2\boldsymbol{x}^{\mathrm{B}}_{t+1} - \left(\boldsymbol{x}^{\mathrm{B}}_{j_k} + \boldsymbol{x}^{\mathrm{B}}_{l_k}\right)\right)$, so adding these together will cancel out the changes.

Proof of Claim 3.3.2. For convenience, we also write

1. $\delta_j \stackrel{\text{def}}{=} \widehat{\boldsymbol{B}}_j \boldsymbol{x}^{\mathrm{B}}$,
2. $\hat{\delta}_i \stackrel{\text{def}}{=} \widehat{\boldsymbol{A}}_i \boldsymbol{x}^{\mathrm{B}} - \boldsymbol{c}^{\mathrm{A}}_i$,
3. $\epsilon_i \stackrel{\text{def}}{=} \boldsymbol{A}_i \boldsymbol{x}^{\mathrm{A}} - \boldsymbol{c}^{\mathrm{A}}_i$.

Thus

$$\left\| \begin{pmatrix} \widehat{\boldsymbol{A}}_i \\ \boldsymbol{W}_i^{1/2} \widehat{\boldsymbol{B}}_i \end{pmatrix} \boldsymbol{x}^{\mathrm{B}} - \begin{pmatrix} \boldsymbol{c}^{\mathrm{A}}_i \\ \boldsymbol{0} \end{pmatrix} \right\|_2^2 = \hat{\delta}_i^2 + w_i \sum_{j \in S_i} \delta_j^2.$$

Summing over all rows, we get

$$\left\| \boldsymbol{B} \begin{pmatrix} \boldsymbol{x}^{\mathrm{A}} \\ \boldsymbol{x}^{\text{aux}} \end{pmatrix} - \boldsymbol{c}^{\mathrm{B}} \right\|_2^2 = \sum_i \left(\hat{\delta}_i^2 + w_i \sum_{j \in S_i} \delta_j^2 \right).$$

Similarly, the square of row i of $\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}} - \boldsymbol{c}^{\mathrm{A}}$ is ϵ_i^2 and summing over all rows we get

$$\left\| \boldsymbol{A}\boldsymbol{x}^{\mathrm{A}} - \boldsymbol{c}^{\mathrm{A}} \right\|_2^2 = \sum_i \epsilon_i^2.$$

Recalling Equation (3.4), we have

$$\boldsymbol{1}^\top \left(\begin{pmatrix} \widehat{\boldsymbol{A}}_i \\ \widehat{\boldsymbol{B}}_i \end{pmatrix} \boldsymbol{x}^{\mathrm{B}} - \begin{pmatrix} \boldsymbol{c}^{\mathrm{A}}_i \\ \boldsymbol{0} \end{pmatrix} \right) = \boldsymbol{A}_i \boldsymbol{x}^{\mathrm{A}} - \boldsymbol{c}^{\mathrm{A}}_i.$$

Thus

$$\hat{\delta}_i + \sum_{j \in S_i} \delta_j = \epsilon_i.$$

By the Cauchy-Schwarz inequality (applied to two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ given by $\boldsymbol{a}_1 = \hat{\delta}_i, \boldsymbol{a}_j = \sqrt{w_i}\delta_j, \boldsymbol{b}_1 = 1, \boldsymbol{b}_j = 1/\sqrt{w_i}$),

$$\epsilon_i^2 = \left(\hat{\delta}_i + \sum_{j \in S_i} \delta_j\right)^2 \leq \left(\hat{\delta}_i^2 + w_i \sum_{j \in S_i} \delta_j^2\right)\left(1 + \frac{m_i}{w_i}\right).$$

the equality holds if and only if

$$\hat{\delta}_i = w_i \delta_j \qquad \forall j \in S_i. \tag{3.5}$$

Note that we have ensured that for all i, $w_i = \alpha m_i$.

By summing over rows we conclude that for every $\boldsymbol{x}^{\mathrm{A}}$ and every $\boldsymbol{x}^{\text{aux}}$, we have

$$\left\|\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}} - \boldsymbol{c}^{\mathrm{A}}\right\|_2^2 \leq \frac{\alpha+1}{\alpha}\left\|\boldsymbol{B}\begin{pmatrix}\boldsymbol{x}^{\mathrm{A}}\\ \boldsymbol{x}^{\text{aux}}\end{pmatrix} - \boldsymbol{c}^{\mathrm{B}}\right\|_2. \tag{3.6}$$

The inequality above will be an equality if Equations (3.5) are satisfied. We now show that for every fixed $\boldsymbol{x}^{\mathrm{A}}$, minimizing over $\boldsymbol{x}^{\text{aux}}$ ensures that (3.6) holds with equality.

In particular, we will momentarily prove the following Claim.

Claim 3.3.4. *For any fixed $\boldsymbol{x}^A$ and its associated ϵ_i values, for each row i of $\boldsymbol{A}$, the linear system*

$$\widehat{\boldsymbol{A}}_i \begin{pmatrix}\boldsymbol{x}^A\\ \boldsymbol{x}^{aux}\end{pmatrix} = \boldsymbol{c}_i^A + \frac{\alpha}{\alpha+1}\epsilon_i, \tag{3.7}$$

$$\widehat{\boldsymbol{B}}_j \begin{pmatrix}\boldsymbol{x}^A\\ \boldsymbol{x}^{aux}\end{pmatrix} = \frac{1}{(\alpha+1)m_i}\epsilon_i, \forall j \in S_i. \tag{3.8}$$

has a solution (which may not be unique).

Since every auxiliary variable is associated with only one row i of $\boldsymbol{A}$, Claim 3.3.4

implies that we can choose $\boldsymbol{x}^{\text{aux}}$ s.t. all these linear systems are satisfied at once.

Given such a choice of $\boldsymbol{x}^{\text{aux}}$, we get that Equations (3.5) are satisfied so

$$\left\|\boldsymbol{A}\boldsymbol{x}^{\text{A}} - \boldsymbol{c}^{\text{A}}\right\|_2^2 = \frac{\alpha+1}{\alpha}\left\|\boldsymbol{B}\begin{pmatrix}\boldsymbol{x}^{\text{A}}\\ \boldsymbol{x}^{\text{aux}}\end{pmatrix} - \boldsymbol{c}^{\text{B}}\right\|_2^2.$$

Given Claim 3.3.4, this completes the proof of Claim 3.3.2. □

Proof of Claim 3.3.4. We focus on the case of the condition $|\mathcal{I}^{+1}| = 1$ AND $|\mathcal{I}^{-1}| = 1$ in Algorithm 1, Line 7 being false. The case when then condition is true is very similar, but easier as it deals with a set of just two equations.

We will construct an assignment to all the variables of $\boldsymbol{x}^{\text{aux}}$ s.t. Equations (3.7) and (3.8) are satisfied. We start with an assignment $\boldsymbol{x}^{\text{A}}$ to the main variables, and we then assign values to auxiliary variables in the order they are created by the algorithm REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{MC}_2$. Note that we will refer to variables j_k and l_k only in the context of a fixed value of t, which always ensures that they are unambiguously defined. When the algorithm processes pair $\boldsymbol{x}^{\text{B}}_{j_k}, \boldsymbol{x}^{\text{B}}_{l_k} = \boldsymbol{u}_{j_k}, \boldsymbol{u}_{l_k}$, the value of these variables will have been set already, while $\boldsymbol{x}^{\text{B}}_t = \boldsymbol{u}_t$ and the other newly created auxiliary variables have not. Every auxiliary variable is associated with only one row, so we never get multiple assignments to a variable using this procedure.

Recall the constraints created by the $\mathcal{MC}_2$Gadget call are

$$\begin{aligned}
\{\boldsymbol{u}_{t+3} - \boldsymbol{v}_{t+3} - (\boldsymbol{u}_{t+4} - \boldsymbol{v}_{t+4}) &= 0,\\
\boldsymbol{u}_t - \boldsymbol{u}_{t+3} &= 0,\\
\boldsymbol{u}_{t+4} - \boldsymbol{u}_{j_k} &= 0,\\
\boldsymbol{v}_{t+3} - \boldsymbol{v}_{t+1} &= 0,\\
\boldsymbol{v}_{t+2} - \boldsymbol{v}_{t+4} &= 0,\\
\boldsymbol{u}_{t+5} - \boldsymbol{v}_{t+5} - (\boldsymbol{u}_{t+6} - \boldsymbol{v}_{t+6}) &= 0,\\
\boldsymbol{u}_t - \boldsymbol{u}_{t+5} &= 0,\\
\boldsymbol{u}_{t+6} - \boldsymbol{u}_{l_k} &= 0,\\
\boldsymbol{v}_{t+5} - \boldsymbol{v}_{t+2} &= 0,\\
\boldsymbol{v}_{t+1} - \boldsymbol{v}_{t+6} &= 0\}
\end{aligned}$$

Let $\boldsymbol{z} = (\boldsymbol{u}_{j_k}, \boldsymbol{u}_{l_k}, \boldsymbol{u}_t, \boldsymbol{u}_{t+1}, \boldsymbol{u}_{t+2}, \boldsymbol{u}_{t+3}, \ldots, \boldsymbol{u}_{t+6}, \boldsymbol{v}_t, \boldsymbol{v}_{t+1}, \ldots, \boldsymbol{v}_{t+6}) \in \mathbb{R}^{16}$. Let $\boldsymbol{G} \in \mathbb{R}^{10\times 16}$ be the matrix s.t. $\boldsymbol{G}\boldsymbol{z} = \boldsymbol{0}$ corresponds to the constraints listed above. Note that all coefficients of $\boldsymbol{u}_{t+1}, \boldsymbol{u}_{t+2}$ and $\boldsymbol{v}_t$ are zero. We set these three variables to zero.

For some ϵ, which we will fix later, we choose $\boldsymbol{u}_t$ such that

$$\boldsymbol{u}_t = \frac{1}{2}\left(\boldsymbol{u}_{j_k} + \boldsymbol{u}_{l_k}\right) + 5\epsilon.$$

Again, for the same ϵ, we fix the following values for $\boldsymbol{u}_{t+i}, \boldsymbol{v}_{t+j}, 3 \leq i \leq 6, 1 \leq j \leq 6$

$$\begin{aligned}
\boldsymbol{u}_{t+3} &= \boldsymbol{u}_t - \epsilon, \\
\boldsymbol{u}_{t+4} &= \boldsymbol{u}_{j_k} + \epsilon, \\
\boldsymbol{u}_{t+5} &= \boldsymbol{u}_t - \epsilon, \\
\boldsymbol{u}_{t+6} &= \boldsymbol{u}_{l_k} + \epsilon, \\
\boldsymbol{v}_{t+1} &= 0, \\
\boldsymbol{v}_{t+2} &= 5\epsilon - (\boldsymbol{u}_t - \boldsymbol{u}_{j_k}), \\
\boldsymbol{v}_{t+3} &= \epsilon, \\
\boldsymbol{v}_{t+4} &= 4\epsilon - (\boldsymbol{u}_t - \boldsymbol{u}_{j_k}), \\
\boldsymbol{v}_{t+5} &= 6\epsilon - (\boldsymbol{u}_t - \boldsymbol{u}_{j_k}), \\
\boldsymbol{v}_{t+6} &= -\epsilon.
\end{aligned}$$

This ensures $\boldsymbol{G}\boldsymbol{z} = \epsilon\mathbf{1}$. Note that for some r, $\widehat{\boldsymbol{B}}_i \boldsymbol{x}^{\mathrm{B}} = 2^r \boldsymbol{G}\boldsymbol{z} = 2^r\epsilon\mathbf{1}$, so by choosing $\epsilon = 2^{-r}\frac{1}{(\alpha+1)m_i}\epsilon_i$, we can ensure Equations (3.8) are satisfied.

Also

$$\boldsymbol{c}_i^{\mathrm{A}} + \epsilon_i = \mathbf{1}^\top \begin{pmatrix} \widehat{\boldsymbol{A}}_i \\ \widehat{\boldsymbol{B}}_i \end{pmatrix} \boldsymbol{x}^{\mathrm{B}} = \widehat{\boldsymbol{A}}_i \boldsymbol{x}^{\mathrm{B}} + \mathbf{1}^\top \widehat{\boldsymbol{B}}_i \boldsymbol{x}^{\mathrm{B}} = \widehat{\boldsymbol{A}}_i \boldsymbol{x}^{\mathrm{B}} + \frac{1}{\alpha+1}\epsilon_i.$$

Thus, we have

$$\widehat{\boldsymbol{A}}_i \boldsymbol{x}^{\mathrm{B}} = \boldsymbol{c}_i^{\mathrm{A}} + \frac{\alpha}{\alpha+1}\epsilon_i,$$

which is Equation 3.7.

This completes the proof of the claim. □

Remark. The optimal solutions for $\min_{\boldsymbol{x}^{\mathrm{B}}} \left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{c}^{\mathrm{B}}\right\|_2$ and $\min_{\boldsymbol{x}^{\mathrm{A}}} \left\|\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}} - \boldsymbol{c}^{\mathrm{A}}\right\|_2$ have a one-to-one map, however, the optimal values are different:

From the proof of Claim 3.3.2 and Equation (2.2)

$$\left\|\boldsymbol{c}^{\mathrm{A}}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2=\left(1+\frac{1}{\alpha}\right)\left\|\boldsymbol{c}^{\mathrm{B}}-\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2^2,$$

where we set the weight $w_i = \alpha m_i, \forall i$. Note when $\alpha \to \infty$, the two optimal values approach the same value.

Relationship Between Schur Complements

Claim 3.3.5. *Let $\boldsymbol{B}$ be the output coefficient matrix of Algorithm 1 with input coefficient matrix $\boldsymbol{A}$ and parameter $\alpha > 0$. Then, $\frac{\alpha}{\alpha+1}\boldsymbol{A}^\top\boldsymbol{A}$ is the Schur complement of $\boldsymbol{B}^\top\boldsymbol{B}$.*

Proof. Write $\boldsymbol{B} = \begin{pmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 \end{pmatrix}$, where $\boldsymbol{B}_1$ is the submatrix corresponding to $\boldsymbol{x}^{\mathrm{A}}$ and $\boldsymbol{B}_2$ is the submatrix corresponding to $\boldsymbol{x}^{\text{aux}}$. Then,

$$\boldsymbol{B}^\top\boldsymbol{B} = \begin{pmatrix} \boldsymbol{B}_1^\top\boldsymbol{B}_1 & \boldsymbol{B}_1^\top\boldsymbol{B}_2 \\ \boldsymbol{B}_2^\top\boldsymbol{B}_1 & \boldsymbol{B}_2^\top\boldsymbol{B}_2 \end{pmatrix}.$$

By Claim 3.3.2, for any fixed $\boldsymbol{x}^{\mathrm{A}}$,

$$\min_{\boldsymbol{x}^{\text{aux}}} \boldsymbol{x}^\top\boldsymbol{B}^\top\boldsymbol{B}\boldsymbol{x} = \frac{\alpha}{\alpha+1}\left(\boldsymbol{x}^{\mathrm{A}}\right)^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}}.$$

By Fact 2.1.2, $\frac{\alpha}{\alpha+1}\boldsymbol{A}^\top\boldsymbol{A}$ is the Schur complement of $\boldsymbol{B}^\top\boldsymbol{B}$. □

3.3.1 Approximate solvers

We now show that approximate solvers for $\boldsymbol{B}$ also translate to approximate solvers for $\boldsymbol{A}$.

Lemma 3.3.6. *Let $\boldsymbol{B}\boldsymbol{x}^B = \boldsymbol{c}^B$ be the linear system returned by a call to* REDUCE $\mathcal{G}_{z,2}$TO$\mathcal{MC}_2(\boldsymbol{A}, \boldsymbol{c}^A, \epsilon^A)$*, and let ϵ^B be the the error parameter returned by this call.*

Let $\boldsymbol{x}^B$ be a vector such that

$$\left\|\boldsymbol{B}\boldsymbol{x}^B - \boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^B\right\|_2 \leq \epsilon^B \left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^B\right\|_2. \tag{3.9}$$

Let $\widetilde{\boldsymbol{x}}^A$ be the vector returned by a call to MAPSOLN $\mathcal{MC}_2$TO$\mathcal{G}_{z,2}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{c}^A, \boldsymbol{x}^B)$*. Then,*

$$\left\|\boldsymbol{A}\widetilde{\boldsymbol{x}}^A - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^A\right\|_2 \leq \epsilon^A \left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^A\right\|_2.$$

Proof. We first consider the case $\boldsymbol{A}^\top \boldsymbol{c}^{\mathrm{A}} = \boldsymbol{0}$, then Algorithm 3 returns $\widetilde{\boldsymbol{x}}^{\mathrm{A}} = \boldsymbol{0}$, which gives

$$\left\|\boldsymbol{A}\widetilde{\boldsymbol{x}}^{\mathrm{A}} - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2 = \boldsymbol{0},$$

and completes the proof for this case. When $\boldsymbol{A}^\top \boldsymbol{c}^{\mathrm{A}} \neq \boldsymbol{0}$, the Algorithm 3 returns $\widetilde{\boldsymbol{x}}^{\mathrm{A}} = \boldsymbol{x}^{\mathrm{A}}$, where $\boldsymbol{x}^{\mathrm{B}} = \begin{pmatrix} \boldsymbol{x}^{\mathrm{A}} \\ \boldsymbol{x}^{\mathrm{aux}} \end{pmatrix}$. Let

$$\boldsymbol{x}^{\mathrm{B}*} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{x}^{\mathrm{A}*} \\ \boldsymbol{x}^{\mathrm{aux}*} \end{pmatrix} \in \arg\min_{\boldsymbol{x}} \left\|\boldsymbol{B}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{B}}\right\|_2.$$

By Fact 2.2.2, the condition (3.9) is equivalent to

$$\left\|\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{x}^{\mathrm{B}*}\right\|_{\boldsymbol{B}^\top \boldsymbol{B}} \leq \epsilon^{\mathrm{B}} \left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2.$$

By Claim 3.3.2, $\boldsymbol{x}^{\mathrm{A}*} \in \arg\min_{\boldsymbol{x}} \left\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{A}}\right\|_2$.

We now upper bound the error of $\left\|\boldsymbol{x}^{\mathrm{A}} - \boldsymbol{x}^{\mathrm{A}*}\right\|_{\boldsymbol{A}^\top \boldsymbol{A}}$. Note

$$\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{x}^{\mathrm{B}*} = \begin{pmatrix} \boldsymbol{x}^{\mathrm{A}} - \boldsymbol{x}^{\mathrm{A}*} \\ \boldsymbol{x}^{\mathrm{aux}} - \boldsymbol{x}^{\mathrm{aux}*} \end{pmatrix}.$$

By Claim 2.1.2 and Claim 3.3.5,

$$\left\|\boldsymbol{x}^{\mathrm{A}}-\boldsymbol{x}^{\mathrm{A}*}\right\|_{\boldsymbol{A}^\top\boldsymbol{A}}^2 \leq \left(1+\frac{1}{\alpha}\right)\left\|\boldsymbol{x}^{\mathrm{B}}-\boldsymbol{x}^{\mathrm{B}*}\right\|_{\boldsymbol{B}^\top\boldsymbol{B}}^2.$$

Then, we lower bound $\left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2$. By Claim 3.3.2,

$$\left\|\boldsymbol{c}^{\mathrm{A}}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 = \left(1+\frac{1}{\alpha}\right)\left\|\boldsymbol{c}^{\mathrm{B}}-\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2^2.$$

Since $\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}} \perp \boldsymbol{c}^{\mathrm{A}}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}$, and $\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}} \perp \boldsymbol{c}^{\mathrm{B}}-\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}$, we have

$$\begin{aligned}\left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 &= \left\|\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 - \left\|\boldsymbol{c}^{\mathrm{A}}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 \\ &= \left\|\boldsymbol{c}^{\mathrm{B}}\right\|_2^2 - \left\|\boldsymbol{c}^{\mathrm{B}}-\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2^2 - \frac{1}{\alpha+1}\left\|\boldsymbol{c}^{\mathrm{A}}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 \\ &= \left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2^2 - \frac{1}{\alpha+1}\left\|\boldsymbol{c}^{\mathrm{A}}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2.\end{aligned}$$

Thus,

$$\left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2^2 = \left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 + \frac{1}{\alpha+1}\left\|\boldsymbol{c}^{\mathrm{A}}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2.$$

Since $\boldsymbol{A}^\top\boldsymbol{c}^{\mathrm{A}} \neq \boldsymbol{0}$, and because $\boldsymbol{A}^\top\boldsymbol{c}^{\mathrm{A}}$ is integral, by Lemma 2.2.5, we have

$$\left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 \geq \frac{1}{\sigma_{\max}^2(\boldsymbol{A})}.$$

Thus,

$$\left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2^2 \leq \left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 + \frac{1}{\alpha+1}\left\|\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 \leq \left(1+\frac{\left\|\boldsymbol{c}^{\mathrm{A}}\right\|_2^2\sigma_{\max}^2(\boldsymbol{A})}{\alpha+1}\right)\left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2.$$

Therefore,

$$\left\|\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2 \leq \left(1+\frac{1}{\alpha}\right)^{1/2}\left(1+\frac{\left\|\boldsymbol{c}^{\mathrm{A}}\right\|_2^2\sigma_{\max}^2(\boldsymbol{A})}{\alpha+1}\right)^{1/2}\epsilon^{\mathrm{B}}\left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2.$$

Finally, by the setting

$$\epsilon^{\mathrm{B}} = \frac{\epsilon^{\mathrm{A}}}{\left(1+\frac{1}{\alpha}\right)^{1/2}\left(1+\frac{\|\boldsymbol{c}^{\mathrm{A}}\|_2^2\sigma_{\max}^2(\boldsymbol{A})}{\alpha+1}\right)^{1/2}},$$

we have

$$\left\|\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}} - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2 \leq \epsilon^{\mathrm{A}}\left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2.$$

This completes the proof. □

3.3.2 Bounding Condition Number of the New Matrix

In this section, we show that the condition number of $\boldsymbol{B}$ is upper bounded by the condition number of $\boldsymbol{A}$ with a poly(n) multiplicative factor.

We first characterize the null space of $\boldsymbol{B}$.

Recall that in Equation (3.1), we write $\boldsymbol{x}^{\mathrm{B}} = \begin{pmatrix} \boldsymbol{x}^{\mathrm{A}} \\ \boldsymbol{x}^{\mathrm{aux}} \end{pmatrix}$. In the following, we will employ a *different indexing* of $\boldsymbol{x}^{\mathrm{aux}}$ than the one defined at the beginning of Section 3.3. We also reorder the columns of $\boldsymbol{B}$ so that $\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}} = \boldsymbol{0}$ represents the same equations as before. For appropriately chosen indices g_1 and g_2, we define

1. $\boldsymbol{x}^{\mathrm{aux}}_{1:g_1}$ corresponds to the $\boldsymbol{u}$-coordinates of the auxiliary variables created in $\mathcal{MC}_2$-gadgets *whose coefficients are nonzero*.

2. $\boldsymbol{x}^{\mathrm{aux}}_{g_1+1:g_2}$ corresponds to the $\boldsymbol{v}$-coordinates of the auxiliary variables created in $\mathcal{MC}_2$-gadgets *whose coefficients are nonzero*.

3. $\boldsymbol{x}^{\mathrm{aux}}_{g_2+1:n_B-n_A}$ corresponds to the coordinates of the auxiliary variables created in $\mathcal{MC}_2$-gadgets *whose coefficients are zero*.

Using this ordering, for $0 \leq i \leq (g_2-g_1)/6$, $\boldsymbol{x}^{\mathrm{aux}}_{g_1+6i+1:g_1+6i+6}$ corresponds to the $\boldsymbol{v}$-coordinates of the auxiliary variables with non-zero coefficients in a single $\mathcal{MC}_2$-gadget.

Given any fixed $\boldsymbol{x}^{\mathrm{A}} \in \mathrm{null}(\boldsymbol{A})$, we extend $\boldsymbol{x}^{\mathrm{A}}$ to a vector in dimension n_B:

$$\boldsymbol{p}(\boldsymbol{x}^{\mathrm{A}}) \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{x}^{\mathrm{A}} \\ \boldsymbol{x}^{\text{aux}} \end{pmatrix}. \tag{3.10}$$

We assign the values of the auxiliary variables of $\boldsymbol{p}(\boldsymbol{x}^{\mathrm{A}})$ in the order that they are created in Algorithm 1. In a $\mathcal{MC}_2$-gadget, suppose the values of variables $\boldsymbol{u}_{j_k}$ and $\boldsymbol{u}_{l_k}$ have already been assigned. Let $\boldsymbol{u}_t, \boldsymbol{u}_{t+i}, \boldsymbol{v}_{t+j}, 3 \leq i \leq 6, 1 \leq j \leq 6$ be the auxiliary variables created in this gadget. We assign values as follows,

$$\begin{aligned}
\boldsymbol{u}_t &= (\boldsymbol{u}_{j_k} + \boldsymbol{u}_{l_k})/2, \\
\boldsymbol{u}_{t+3} &= \boldsymbol{u}_t. \\
\boldsymbol{u}_{t+4} &= \boldsymbol{u}_{j_k}, \\
\boldsymbol{u}_{t+5} &= \boldsymbol{u}_t, \\
\boldsymbol{u}_{t+6} &= \boldsymbol{u}_{l_k}, \\
\boldsymbol{v}_{t+1} &= 0, \\
\boldsymbol{v}_{t+2} &= \boldsymbol{u}_t - \boldsymbol{u}_{j_k}, \\
\boldsymbol{v}_{t+3} &= 0, \\
\boldsymbol{v}_{t+4} &= \boldsymbol{u}_t - \boldsymbol{u}_{j_k}, \\
\boldsymbol{v}_{t+5} &= \boldsymbol{u}_t - \boldsymbol{u}_{j_k}, \\
\boldsymbol{v}_{t+6} &= 0.
\end{aligned}$$

This gives the first g_2 entries of $\boldsymbol{x}^{\text{aux}}$, and we set all the rest of the entries to be 0.

Let $\boldsymbol{e}_i \in \mathbb{R}^{(g_2-g_1)/6}$ be the ith standard basis vector and $\mathbf{1}$ be the all-one vector in 6 dimensions. We define $\boldsymbol{p}^i \in \mathbb{R}^{n_B}$ to be a vector whose nonzero entries are given by

$$\boldsymbol{p}^i_{n_A+g_1+1:n_A+g_2} = \boldsymbol{e}_i \otimes \mathbf{1}. \tag{3.11}$$

Let $\boldsymbol{e}_j \in \mathbb{R}^{n_B - n_A - g_2}$ is the jth standard basis vector. We define $\boldsymbol{q}^j \in \mathbb{R}^{n_B}$ to be a vector whose nonzero entries are given by

$$\boldsymbol{q}^j_{n_A+g_2+1:n_B} = \boldsymbol{e}_j. \tag{3.12}$$

Lemma 3.3.7. $\operatorname{null}(\boldsymbol{B}) = \textbf{\textit{Span}}\left(\boldsymbol{p}(\boldsymbol{x}^A), \boldsymbol{p}^i, \boldsymbol{q}^j : \boldsymbol{x}^A \in \operatorname{null}(\boldsymbol{A}), 1 \le i \le \frac{g_2-g_1}{6}, 1 \le j \le n_B - g_2\right)$.

Proof. Let

$$\boldsymbol{S} \stackrel{\text{def}}{=} \textbf{Span}\left(\boldsymbol{p}(\boldsymbol{x}^{\mathrm{A}}), \boldsymbol{p}^i, \boldsymbol{q}^j : \boldsymbol{x}^{\mathrm{A}} \in \operatorname{null}(\boldsymbol{A}), 1 \le i \le \frac{g_2 - g_1}{6}, 1 \le j \le n_B - g_2\right).$$

According to definitions of the vectors, we can check that $\boldsymbol{S} \subseteq \operatorname{null}(\boldsymbol{B})$.

It remains to show that $\boldsymbol{S} \supseteq \operatorname{null}(\boldsymbol{B})$. Let $\boldsymbol{x} \in \operatorname{null}(\boldsymbol{B})$. By Claim 3.3.2 with $\boldsymbol{c}^{\mathrm{A}} = \boldsymbol{0}$, we have

$$\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}} = \boldsymbol{0},$$

that is, $\boldsymbol{x}^{\mathrm{A}} \in \operatorname{null}(\boldsymbol{A})$. According to the $\mathcal{MC}_2$-gadget constraints in Algorithm 2, we have

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_{t+3} = \boldsymbol{v}_{t+6} = \gamma, \text{ and } \boldsymbol{v}_{t+2} = \boldsymbol{v}_{t+4} = \boldsymbol{v}_{t+5} = \theta,$$

where $\gamma - \theta = (\boldsymbol{u}_{j_k} - \boldsymbol{u}_{j_l})/2$. Besides, since all entries in $\boldsymbol{x}_{n_A+g_2+1:n_B}$ have zero coefficients, they are free to choose. Thus, $\boldsymbol{x} \in \boldsymbol{S}$, that is, $\boldsymbol{S} \supseteq \operatorname{null}(\boldsymbol{B})$.

This completes the proof. □

We upper bound the largest singular value of $\boldsymbol{B}$ in the following claim.

Lemma 3.3.8. $\lambda_{\max}\left(\boldsymbol{B}^\top \boldsymbol{B}\right) = O\left(\alpha \operatorname{nnz}(\boldsymbol{A})^2 \|\boldsymbol{A}\|_\infty^2 \log^2 \|\boldsymbol{A}\|_\infty\right)$.

Proof. By the Courant-Fischer Theorem,

$$\lambda_{\max}\left(\boldsymbol{B}^\top \boldsymbol{B}\right) = \max_{\boldsymbol{x}} \frac{\boldsymbol{x}^\top \boldsymbol{B}^\top \boldsymbol{B} \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}}.$$

We write $\boldsymbol{x}$ as $(\boldsymbol{u}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{u}_{n_B}, \boldsymbol{v}_{n_B})$, so that each $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ corresponds to the two coordinates of vertex i in the graph. Expanding the right hand side, we get

$$\frac{\sum_{(i,j)\in E_1} w_{(i,j)}(\boldsymbol{u}_i - \boldsymbol{u}_j)^2 + \sum_{(i,j)\in E_2} w_{(i,j)}(\boldsymbol{v}_i - \boldsymbol{v}_j)^2 + \sum_{(i,j)\in E_{1+2}} w_{(i,j)}(\boldsymbol{u}_i - \boldsymbol{v}_i - (\boldsymbol{u}_j - \boldsymbol{v}_j))^2}{\sum_i \boldsymbol{u}_i^2 + \boldsymbol{v}_i^2}$$

where $w_{(i,j)}$'s are the edge weights. Let

$$w_{\max} = \max_{(i,j)\in E_1 \cup E_2 \cup E_{1+2}} w_{(i,j)}.$$

According to our construction of $\boldsymbol{B}$ in Algorithm 1,

$$w_{\max} \leq \alpha \operatorname{nnz}(\boldsymbol{B}) \left\| \boldsymbol{A} \right\|_\infty^2. \tag{3.13}$$

Thus, $\lambda_{\max}\left(\boldsymbol{B}^\top \boldsymbol{B}\right)$ is at most

$$w_{\max} \max_{\boldsymbol{u}} \frac{\sum_{(i,j)\in E_1} (\boldsymbol{u}_i - \boldsymbol{u}_j)^2 + \sum_{(i,j)\in E_2} (\boldsymbol{v}_i - \boldsymbol{v}_j)^2 + \sum_{(i,j)\in E_{1+2}} (\boldsymbol{u}_i - \boldsymbol{v}_i - (\boldsymbol{u}_j - \boldsymbol{v}_j))^2}{\sum_i \boldsymbol{u}_i^2 + \boldsymbol{v}_i^2}.$$

We upper bound each term of the numerator. Let d_1, d_2, d_{1+2} be the maximum vertex degree of the graphs G_1, G_2, G_{1+2}, respectively. By Cauchy-Schwarz inequality,

$$\sum_{(i,j)\in E_1} (\boldsymbol{u}_i - \boldsymbol{u}_j)^2 \leq 2 \sum_{(i,j)\in E_1} \boldsymbol{u}_i^2 + \boldsymbol{u}_j^2 \leq 2d_1 \left(\sum_i \boldsymbol{u}_i^2 + \boldsymbol{v}_i^2 \right),$$

$$\sum_{(i,j)\in E_2} (\boldsymbol{v}_i - \boldsymbol{v}_j)^2 \leq 2d_2 \left(\sum_i \boldsymbol{u}_i^2 + \boldsymbol{v}_i^2 \right),$$

and

$$\sum_{(i,j)\in E_{1+2}} (\boldsymbol{u}_i - \boldsymbol{v}_i - (\boldsymbol{u}_j - \boldsymbol{v}_j))^2 \leq 4 \sum_{(i,j)\in E_{1+2}} \boldsymbol{u}_i^2 + \boldsymbol{v}_i^2 + \boldsymbol{u}_j^2 + \boldsymbol{v}_j^2$$
$$\leq 4d_{1+2} \left(\sum_i \boldsymbol{u}_i^2 + \boldsymbol{v}_i^2 \right).$$

Plugging the above inequalities to the expansion, we have

$$\lambda_{\max}\left(\boldsymbol{B}^\top \boldsymbol{B}\right) \leq w_{\max} \cdot 8 \max\{d_1, d_2, d_{1+2}\}$$
$$\leq 8 w_{\max} \operatorname{nnz}(\boldsymbol{B}).$$

By the upper bound of $\operatorname{nnz}(\boldsymbol{B})$ in Lemma 3.3.1 and the upper bound of $w_{\max}$ in Equation (3.13), we have

$$\lambda_{\max}\left(\boldsymbol{B}^\top \boldsymbol{B}\right) \leq O\left(\alpha \operatorname{nnz}(\boldsymbol{A})^2 \left\|\boldsymbol{A}\right\|_\infty^2 \log^2 \left\|\boldsymbol{A}\right\|_\infty\right).$$

This completes the proof. □

Recall that, for a vector $\boldsymbol{x} \in \mathbb{R}^{n_B}$, $\boldsymbol{x}_{n_A+1:n_A+g_1}$ corresponds to the auxiliary $\boldsymbol{u}$-variables with non-zero coefficients in $\mathcal{MC}_2$-gadgets, $\boldsymbol{x}_{n_A+1+g_1+6i:n_A+6+g_1+6i}$ for $0 \leq i \leq (g_2-g_1)/6$ corresponds to the auxiliary $\boldsymbol{v}$-variables with non-zero coefficients in a single $\mathcal{MC}_2$-gadget, and $\boldsymbol{x}_{n+g_2+1:n_B}$ corresponds to the variables with zero coefficient.

Lemma 3.3.9. *Let* $0 \leq \epsilon \leq (100 n_B \log \left\|\boldsymbol{A}\right\|_\infty)^{-1}$. *Let* $\boldsymbol{x}^B \in \mathbb{R}^{n_B}$ *satisfying*

1. $\boldsymbol{x}^B_{n_A+g_2+1:n_B} = \boldsymbol{0}$,
2. $\boldsymbol{x}^B_{n_A+g_1+6i+1} = 0, \forall 0 \leq i \leq (g_2 - g_1)/6$, *and*
3. $\left\|\boldsymbol{B}\boldsymbol{x}^B\right\|_2 \leq \epsilon \left\|\boldsymbol{x}^B\right\|_2$.

Then,

$$\left\|\boldsymbol{A}\boldsymbol{x}^A\right\|_2 \leq 4 \left(1 + \frac{1}{\alpha}\right)^{1/2} n_B^{1/2} \epsilon \left\|\boldsymbol{x}^A\right\|_2.$$

Proof. We first show that under the conditions of the Lemma,

$$\left\|\boldsymbol{x}^{\mathrm{A}}\right\|_\infty \geq \frac{1}{4}\left\|\boldsymbol{x}^{\mathrm{B}}\right\|_\infty . \tag{3.14}$$

Let $\delta \stackrel{\text{def}}{=} \epsilon \left\|\boldsymbol{x}^{\mathrm{B}}\right\|_2$. By the 3rd condition, each entry of $\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}}$ has absolute value at most δ.

We first show that all $\boldsymbol{u}$-variables in $\boldsymbol{x}^{\text{aux}}$ cannot be large. According to Algorithm 1, for each row of $\boldsymbol{B}$, all nonzero coefficients have same absolute value, which is at least 1. Based on this fact and the $\mathcal{MC}_2$-gadget constructed in Algorithm 2, we have

$$\begin{aligned}
\frac{\boldsymbol{u}_{j_k}+\boldsymbol{u}_{l_k}}{2} - 5\delta \leq \boldsymbol{u}_t \leq \frac{\boldsymbol{u}_{j_k}+\boldsymbol{u}_{l_k}}{2} + 5\delta \\
\frac{\boldsymbol{u}_{j_k}+\boldsymbol{u}_{l_k}}{2} - 6\delta \leq \boldsymbol{u}_{t+3} \leq \frac{\boldsymbol{u}_{j_k}+\boldsymbol{u}_{l_k}}{2} + 6\delta \\
\boldsymbol{u}_{j_k} - \delta \leq \boldsymbol{u}_{t+4} \leq \boldsymbol{u}_{j_k} + \delta \\
\frac{\boldsymbol{u}_{j_k}+\boldsymbol{u}_{l_k}}{2} - 6\delta \leq \boldsymbol{u}_{t+5} \leq \frac{\boldsymbol{u}_{j_k}+\boldsymbol{u}_{l_k}}{2} + 6\delta \\
\boldsymbol{u}_{l_k} - \delta \leq \boldsymbol{u}_{t+6} \leq \boldsymbol{u}_{l_k} + \delta
\end{aligned}$$

where $\boldsymbol{u}_{j_k}, \boldsymbol{u}_{l_k}$ being paired-and-replaced, and all others are entries of $\boldsymbol{x}^{\text{aux}}$. By the triangle inequality,

$$\left|\boldsymbol{u}_t\right|, \left|\boldsymbol{u}_{t+k}\right| \leq \frac{1}{2}\left(\left|\boldsymbol{u}_{j_k}\right| + \left|\boldsymbol{u}_{l_k}\right|\right) + 6\delta, \quad \forall 3 \leq k \leq 6.$$

Note that the sum of coefficients on both sides are equal. We can repeat this type of substitution on the right hand side until $\boldsymbol{u}_{j_k}$ and $\boldsymbol{u}_{l_k}$ are variables of $\boldsymbol{x}^{\mathrm{A}}$. At the ith iteration of Algorithm 1 line 20, we have

$$\left|\boldsymbol{u}_t\right|, \left|\boldsymbol{u}_{t+k}\right| \leq \sum_j \alpha_j \left|\boldsymbol{x}^{\mathrm{A}}_j\right| + 6r\delta, \qquad \forall 3 \leq k \leq 6,$$

where $\alpha_j \geq 0$ and $\sum_j \alpha_j = 1$. By the Hölder inequality,

$$|\boldsymbol{u}_t|, |\boldsymbol{u}_{t+k}| \leq \left\|\boldsymbol{x}^{\mathrm{A}}\right\|_\infty + 6r\delta, \qquad \forall 3 \leq k \leq 6. \tag{3.15}$$

We then argue that all $\boldsymbol{v}$-variables in $\boldsymbol{x}^{\text{aux}}$ cannot be large. Note that by the 2nd condition, in the $\mathcal{MC}_2$-gadget, we have $\boldsymbol{v}_{t+1} = 0$. According to the equations in Algorithm 2, we have

$$\begin{gathered}
\boldsymbol{u}_{t+3} - \boldsymbol{u}_{t+4} - 3\delta \leq \boldsymbol{v}_{t+2} \leq \boldsymbol{u}_{t+3} - \boldsymbol{u}_{t+4} + 3\delta \\
-\delta \leq \boldsymbol{v}_{t+3} \leq \delta \\
\boldsymbol{u}_{t+3} - \boldsymbol{u}_{t+4} - 2\delta \leq \boldsymbol{v}_{t+4} \leq \boldsymbol{u}_{t+3} - \boldsymbol{u}_{t+4} + 2\delta \\
\boldsymbol{u}_{t+3} - \boldsymbol{u}_{t+4} - 4\delta \leq \boldsymbol{v}_{t+5} \leq \boldsymbol{u}_{t+3} - \boldsymbol{u}_{t+4} + 4\delta \\
-\delta \leq \boldsymbol{v}_{t+6} \leq \delta
\end{gathered}$$

By the triangle inequality,

$$|\boldsymbol{v}_{t+k}| \leq |\boldsymbol{u}_{t+3}| + |\boldsymbol{u}_{t+4}| + 4\delta, \qquad \forall 2 \leq k \leq 6.$$

By (3.15), at the rth iteration of Algorithm 1 line 20,

$$|\boldsymbol{v}_{t+k}| \leq 2\left\|\boldsymbol{x}^{\mathrm{A}}\right\|_\infty + 12r\delta, \qquad \forall 2 \leq k \leq 6.$$

Since there are at most $\log \|\boldsymbol{A}\|_\infty$ iterations, the above inequality together with Equation (3.15) implies

$$\|\boldsymbol{x}^{\text{aux}}\|_\infty \leq 2\left\|\boldsymbol{x}^{\mathrm{A}}\right\|_\infty + 12\delta \log \|\boldsymbol{A}\|_\infty .$$

Adding $\left\|\boldsymbol{x}^{\mathrm{A}}\right\|_\infty$ on both sides and substituting $\delta = \epsilon \left\|\boldsymbol{x}^{\mathrm{B}}\right\|_2$ gives

$$\begin{aligned}\left\|\boldsymbol{x}^{\mathrm{B}}\right\|_\infty &\leq 3\left\|\boldsymbol{x}^{\mathrm{A}}\right\|_\infty + 12\epsilon\left\|\boldsymbol{x}^{\mathrm{B}}\right\|_2 \log\|\boldsymbol{A}\|_\infty \\ &\leq 3\left\|\boldsymbol{x}^{\mathrm{A}}\right\|_\infty + 12\epsilon\sqrt{n_B}\left\|\boldsymbol{x}^{\mathrm{B}}\right\|_\infty \log\|\boldsymbol{A}\|_\infty .\end{aligned}$$

Given $\epsilon \leq \left(100\sqrt{n_B}\log\|\boldsymbol{A}\|_\infty\right)^{-1}$, we have Equation (3.14). This further implies

$$\left\|\boldsymbol{x}^{\mathrm{B}}\right\|_2 \leq 4\sqrt{n_B}\left\|\boldsymbol{x}^{\mathrm{A}}\right\|_2 . \tag{3.16}$$

By Claim 3.3.2 with $\boldsymbol{c}^{\mathrm{B}} = \boldsymbol{0}$, we have

$$\left\|\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}}\right\|_2 \leq \sqrt{\frac{\alpha+1}{\alpha}}\left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}}\right\|_2 .$$

By the 3rd condition and the bound (3.16), we have

$$\left\|\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}}\right\|_2 \leq 4\left(1+\frac{1}{\alpha}\right)^{1/2} n_B^{1/2}\epsilon\left\|\boldsymbol{x}^{\mathrm{A}}\right\|_2 .$$

This completes the proof. □

We use the above lemma to lower bound the smallest nonzero eigenvalue of $\boldsymbol{B}^\top\boldsymbol{B}$.

Lemma 3.3.10. $\lambda_{\min}\left(\boldsymbol{B}^\top\boldsymbol{B}\right) = \Omega\left(\frac{\min\{1,\lambda_{\min}(\boldsymbol{A}^\top\boldsymbol{A})\}}{n_B^2}\right)$.

Proof. Let

$$\delta \stackrel{\text{def}}{=} \min\{1, \lambda_{\min}(\boldsymbol{A}^\top\boldsymbol{A})\}, \tag{3.17}$$

and

$$\epsilon \stackrel{\text{def}}{=} \frac{\delta}{1616(1+\alpha^{-1})n_B^2}. \tag{3.18}$$

The goal is to prove that $\lambda_{\min}(\boldsymbol{B}^\top \boldsymbol{B}) \geq \epsilon$. Assume by contradiction, there exists an $\boldsymbol{x} \perp \mathrm{null}(\boldsymbol{B})$ such that

$$(\boldsymbol{x}^{\mathrm{B}})^\top \boldsymbol{B}^\top \boldsymbol{B} \boldsymbol{x}^{\mathrm{B}} < \epsilon (\boldsymbol{x}^{\mathrm{B}})^\top \boldsymbol{x}^{\mathrm{B}}. \tag{3.19}$$

We show a contradiction by case analysis according to whether $\boldsymbol{x}^{\mathrm{A}}$ is orthogonal to the null space of $\boldsymbol{A}$.

Case 1: Suppose $\boldsymbol{x}^{\mathrm{A}} \perp \mathrm{null}(\boldsymbol{A})$.

Recall that $\boldsymbol{x}^{\mathrm{B}}_{n_A+1:n_A+g_1}$ corresponds to the auxiliary $\boldsymbol{u}$-variables in the constraints, $\boldsymbol{x}^{\mathrm{B}}_{n_A+g_1+1:n_A+g_2}$ corresponds to the auxiliary $\boldsymbol{v}$-variables in the constraints, and $\boldsymbol{x}^{\mathrm{B}}_{n_A+g_2+1:n_B}$ corresponds to the variables with zero coefficient. By Lemma 3.3.7, we have $\boldsymbol{x}^{\mathrm{B}} \perp \boldsymbol{q}^j, \forall 1 \leq j \leq n_A - g_2$, where $\boldsymbol{q}^j$ is defined in Equation (3.12). Thus, $\boldsymbol{x}_{n_A+g_2+1:n_B} = \boldsymbol{0}$.

Let $\boldsymbol{r} \stackrel{\text{def}}{=} \sum_{0 \leq i \leq (g_2-g_1)/6} \boldsymbol{x}^{\mathrm{B}}_{n_A+g_1+6i+1} \boldsymbol{p}^i$, where $\boldsymbol{p}^i$ is defined in Equation (3.11). By Lemma 3.3.7, $\boldsymbol{r} \in \mathrm{null}(\boldsymbol{B})$. Let $\boldsymbol{y}^{\mathrm{B}} \stackrel{\text{def}}{=} \boldsymbol{x}^{\mathrm{B}} - \boldsymbol{r}$. Similarly, we write

$$\boldsymbol{y}^{\mathrm{B}} = \begin{pmatrix} \boldsymbol{y}^{\mathrm{A}} \\ \boldsymbol{y}^{\mathrm{aux}} \end{pmatrix},$$

where $\boldsymbol{y}^{\mathrm{A}}$ corresponds to original variables and $\boldsymbol{y}^{\mathrm{aux}}$ corresponds to auxiliary variables. Note $\boldsymbol{y}^{\mathrm{A}} = \boldsymbol{x}^{\mathrm{A}}$. Since $\boldsymbol{r} \in \mathrm{null}(\boldsymbol{B})$, we have

$$(\boldsymbol{y}^{\mathrm{B}})^\top \boldsymbol{B}^\top \boldsymbol{B} \boldsymbol{y}^{\mathrm{B}} = (\boldsymbol{x}^{\mathrm{B}})^\top \boldsymbol{B}^\top \boldsymbol{B} \boldsymbol{x}^{\mathrm{B}}.$$

Since $\boldsymbol{x}^{\mathrm{B}} \perp \boldsymbol{r}$, we have

$$(\boldsymbol{y}^{\mathrm{B}})^\top \boldsymbol{y}^{\mathrm{B}} = (\boldsymbol{x}^{\mathrm{B}})^\top \boldsymbol{x}^{\mathrm{B}} + \boldsymbol{r}^\top \boldsymbol{r} \geq (\boldsymbol{x}^{\mathrm{B}})^\top \boldsymbol{x}^{\mathrm{B}}.$$

By the assumption in Equation (3.19), we have

$$(\boldsymbol{y}^{\mathrm{B}})^{\top}\boldsymbol{B}^{\top}\boldsymbol{B}\boldsymbol{y}^{\mathrm{B}} < \epsilon(\boldsymbol{y}^{\mathrm{B}})^{\top}\boldsymbol{y}^{\mathrm{B}}.$$

By our definition of $\boldsymbol{y}^{\mathrm{B}}$, $\boldsymbol{y}^{\mathrm{B}}$ satisfies the conditions of Lemma 3.3.9. By Lemma 3.3.9 and the definition of ϵ in Equation (3.18), we have

$$\left(\boldsymbol{x}^{\mathrm{A}}\right)^{\top}\boldsymbol{A}^{\top}\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}} < \delta\left(\boldsymbol{x}^{\mathrm{A}}\right)^{\top}\boldsymbol{x}^{\mathrm{A}}.$$

Since $\boldsymbol{x}^{\mathrm{A}} \perp \mathrm{null}(\boldsymbol{A})$ and the definition of δ in Equation (3.17), we have

$$\lambda_{\min}\left(\boldsymbol{A}^{\top}\boldsymbol{A}\right) < \delta = \min\left\{1, \lambda_{\min}(\boldsymbol{A}^{\top}\boldsymbol{A})\right\},$$

which is a contradiction.

Case 2: Suppose $\boldsymbol{x}^{\mathrm{A}} \in \mathrm{null}(\boldsymbol{A})$.

Recall that $\boldsymbol{p}(\boldsymbol{x}^{\mathrm{A}})$ is the extension of vector $\boldsymbol{x}^{\mathrm{A}}$ defined in Equation (3.10). Let $\boldsymbol{r} = \sum_{0\le i\le(g_2-g_1)/6}\alpha_i\boldsymbol{p}^i$ be a vector such that

$$\boldsymbol{y}^{\mathrm{B}} \stackrel{\mathrm{def}}{=} \boldsymbol{x}^{\mathrm{B}} - \boldsymbol{p}(\boldsymbol{x}^{\mathrm{A}}) - \boldsymbol{r}$$

satisfying $\boldsymbol{y}^{\mathrm{B}}_{n_A+g_1+6i+1} = \boldsymbol{0}, \forall 0 \le i < (g_2-g_1)/6$. By this definition, we have

$$\boldsymbol{y}^{\mathrm{A}} = \boldsymbol{0}. \tag{3.20}$$

Since $\boldsymbol{p}(\boldsymbol{x}^{\mathrm{A}}) + \boldsymbol{r}$ is in the null space of $\boldsymbol{B}$, by the assumption in Equation (3.19),

$$(\boldsymbol{y}^{\mathrm{B}})^{\top}\boldsymbol{B}^{\top}\boldsymbol{B}\boldsymbol{y}^{\mathrm{B}} < \epsilon(\boldsymbol{y}^{\mathrm{B}})^{\top}\boldsymbol{y}^{\mathrm{B}}.$$

Note $\boldsymbol{y}^{\mathrm{B}}$ satisfies the conditions of Lemma 3.3.9. By Equations (3.14) and (3.20), we have

$$\left\|\boldsymbol{y}^{\mathrm{B}}\right\|_{\infty} = 0.$$

This implies that $\boldsymbol{x}^{\mathrm{B}} = \boldsymbol{p}(\boldsymbol{x}^{\mathrm{A}}) + \boldsymbol{r}$ is in the null space of $\boldsymbol{B}$, which contradicts that $\boldsymbol{x}^{\mathrm{B}} \perp \mathrm{null}(\boldsymbol{B})$.

Case 3: Suppose $\boldsymbol{x}^{\mathrm{A}} = \tilde{\boldsymbol{z}} + \hat{\boldsymbol{z}}$, where $\tilde{\boldsymbol{z}} \neq \mathbf{0}$ is in $\mathrm{null}(\boldsymbol{A})$ and $\hat{\boldsymbol{z}} \neq \mathbf{0}$ is orthogonal to $\mathrm{null}(\boldsymbol{A})$.

Similarly, let $\boldsymbol{r}_1 = \sum_i \alpha_i \boldsymbol{p}^i$ such that $\boldsymbol{y}^{\mathrm{B}} \stackrel{\mathrm{def}}{=} \boldsymbol{x}^{\mathrm{B}} - \boldsymbol{r}_1$ satisfying $\boldsymbol{y}^{\mathrm{B}}_{n_A+g_1+6i+1} = 0, \forall i$. By this definition, we have

$$\boldsymbol{y}^{\mathrm{A}} = \boldsymbol{x}^{\mathrm{A}}.$$

Then by the assumption in Equation (3.19),

$$(\boldsymbol{y}^{\mathrm{B}})^{\top}\boldsymbol{B}^{\top}\boldsymbol{B}\boldsymbol{y}^{\mathrm{B}} < \epsilon(\boldsymbol{y}^{\mathrm{B}})^{\top}\boldsymbol{y}^{\mathrm{B}}.$$

$\boldsymbol{y}^{\mathrm{B}}$ satisfies the conditions of Lemma 3.3.9. By Lemma 3.3.9, we have

$$(\boldsymbol{y}^{\mathrm{A}})^{\top}\boldsymbol{A}^{\top}\boldsymbol{A}\boldsymbol{y}^{\mathrm{A}} \leq 16\left(1+\frac{1}{\alpha}\right)n_B\epsilon(\boldsymbol{y}^{\mathrm{A}})^{\top}\boldsymbol{y}^{\mathrm{A}}.$$

Since $\boldsymbol{y}^{\mathrm{A}} = \tilde{\boldsymbol{z}} + \hat{\boldsymbol{z}}$ with $\tilde{\boldsymbol{z}} \in \mathrm{null}(\boldsymbol{A})$ and $\hat{\boldsymbol{z}} \perp \mathrm{null}(\boldsymbol{A})$, we have

$$\hat{\boldsymbol{z}}^{\top}\boldsymbol{A}^{\top}\boldsymbol{A}\hat{\boldsymbol{z}} \leq 16\left(1+\frac{1}{\alpha}\right)n_B\epsilon\left(\tilde{\boldsymbol{z}}^{\top}\tilde{\boldsymbol{z}} + \hat{\boldsymbol{z}}^{\top}\hat{\boldsymbol{z}}\right).$$

If $\hat{\boldsymbol{z}}^{\top}\hat{\boldsymbol{z}} > \tilde{\boldsymbol{z}}^{\top}\tilde{\boldsymbol{z}}/(100n_B)$, then we have a contradiction with Equation (3.17) and we have done. Otherwise, we have

$$\|\hat{\boldsymbol{z}}\|_2^2 \leq \frac{\|\tilde{\boldsymbol{z}}\|_2^2}{100n_B}.$$

Let $\boldsymbol{r}_2 = \sum_i \beta_i \boldsymbol{p}^i$ such that

$$\boldsymbol{z}^{\mathrm{B}} \stackrel{\mathrm{def}}{=} \boldsymbol{y}^{\mathrm{B}} - \boldsymbol{p}(\tilde{\boldsymbol{z}}) - \boldsymbol{r}_2$$

satisfies $\boldsymbol{z}^{\mathrm{B}}_{n_A+g_1+6i+1} = 0, \forall i$. By the assumption in Equation (3.19),

$$(\boldsymbol{z}^{\mathrm{B}})^\top \boldsymbol{B}^\top \boldsymbol{B} \boldsymbol{z}^{\mathrm{B}} < \epsilon (\boldsymbol{z}^{\mathrm{B}})^\top \boldsymbol{z}^{\mathrm{B}}.$$

Vector $\boldsymbol{z}^{\mathrm{B}}$ satisfies the conditions of Lemma 3.3.9. By Equation (3.14), we have

$$\left\|\boldsymbol{z}^{\mathrm{B}}\right\|_2 \leq 4\sqrt{n_B} \left\|\hat{\boldsymbol{z}}\right\|_2.$$

Since $\hat{\boldsymbol{z}}^\top \hat{\boldsymbol{z}} \leq \tilde{\boldsymbol{z}}^\top \tilde{\boldsymbol{z}}/(100 n_B)$, we have

$$\left\|\boldsymbol{z}^{\mathrm{B}}\right\|_2 \leq \left\|\tilde{\boldsymbol{z}}\right\|_2.$$

On the other hand,

$$\begin{aligned}
\left\|\boldsymbol{z}^{\mathrm{B}}\right\|_2^2 &= \left\|\boldsymbol{x}^{\mathrm{B}}\right\|_2^2 + \left\|\boldsymbol{p}(\tilde{\boldsymbol{z}}) + \boldsymbol{r}_1 + \boldsymbol{r}_2\right\|_2^2 \\
&> \left\|\boldsymbol{p}(\tilde{\boldsymbol{z}}) + \boldsymbol{r}_1 + \boldsymbol{r}_2\right\|_2^2 \\
&\geq \left\|\boldsymbol{x}^{\mathrm{A}}\right\|_2^2 \\
&= \left\|\tilde{\boldsymbol{z}}\right\|_2^2 + \left\|\hat{\boldsymbol{z}}\right\|_2^2 \\
&> \left\|\tilde{\boldsymbol{z}}\right\|_2^2.
\end{aligned}$$

The first equality is due to $\boldsymbol{x}^{\mathrm{B}} \perp (\boldsymbol{p}(\tilde{\boldsymbol{z}}) + \boldsymbol{r}_1 + \boldsymbol{r}_2)$, the third inequality is due to the $\boldsymbol{x}^{\mathrm{A}}$ part of $\boldsymbol{r}_1 + \boldsymbol{r}_2$ is $\mathbf{0}$, and the fourth equality is due to that $\tilde{\boldsymbol{z}} \perp \hat{\boldsymbol{z}}$. Thus, we get a contradiction.

This completes the proof. □

Lemma 3.3.8 and Lemma 3.3.10 immediately imply the following lemma.

Lemma 3.3.11. $\kappa(\boldsymbol{B}) = O\left(\frac{\mathrm{nnz}(A)^2 \|\boldsymbol{A}\|_\infty \log^2 \|\boldsymbol{A}\|_\infty}{\min\{1, \sigma_{\min}(\boldsymbol{A})\}}\right)$.

3.3.3 Putting it All Together

In this section, we prove Lemma 3.2.5.

Proof of Lemma 3.2.5. We set $\alpha = 1$ in Algorithm 1.

By Lemma 3.3.1 and Claim 3.1.2, we have

$$\mathrm{nnz}(\boldsymbol{B}) = O\left(s \log(sU)\right).$$

According to our reduction in Algorithm 1, the largest entry and the smallest nonzero entry of the right hand side vector does not change. Besides, all nonzero entries of $\boldsymbol{B}$ have absolute value at least 1. The largest entry of $\boldsymbol{B}$ is upper bounded by Equation (3.13)

$$\sqrt{w_{\max}} = O\left(\sqrt{\mathrm{nnz}(\boldsymbol{B})}\,\|\boldsymbol{A}\|_\infty\right).$$

By Claim 3.1.2, we have

$$\sqrt{w_{\max}} = O\left(s^{3/2} U \log^{1/2}(sU)\right).$$

By Lemma 3.3.11 and Lemma 3.1.2, 3.1.2, we have

$$\kappa(\boldsymbol{B}) = O\left(\frac{\mathrm{nnz}(A)^2\,\|\boldsymbol{A}\|_\infty \log^2 \|\boldsymbol{A}\|_\infty}{\min\{1, \sigma_{\min}(\boldsymbol{A})\}}\right) = O\left(s^4 U^2 K \log^2(sU)\right).$$

By Lemma 3.3.6, we have

$$(\epsilon^{\mathrm{B}})^{-1} = (\epsilon^{\mathrm{A}})^{-1} O\left(1 + \left\|\boldsymbol{c}^{\mathrm{A}}\right\|_2 \sigma_{\max}(\boldsymbol{A})\right) = (\epsilon^{\mathrm{A}})^{-1} O\left(sU^2\right).$$

This completes the proof. □

3.4 $\mathcal{MC}_2$ Efficiently Reducible to $\mathcal{MC}_2^{>0}$

The matrices generated by interior point methods (see Section 3.8 for details) are more restrictive than $\mathcal{MC}_2$: for every edge present, the weights of all three types of matrices are non-zero. Formally, the class of matrices $\mathcal{MC}_2^{>0}$ consists of matrices of the form

$$\boldsymbol{L}^1 \otimes \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \boldsymbol{L}^2 \otimes \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} + \boldsymbol{L}^{1+2} \otimes \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},$$

where the non-zero support of all three matrices, $\boldsymbol{L}^1$, $\boldsymbol{L}^2$, and $\boldsymbol{L}^{1+2}$ are the same (but the weights may vary greatly). On the other hand, the matrices that we generate in Section 3.3 can be transformed into such a matrix by adding a small value, δ to the weight of all edges where one of the types of edges have non-zero support. We will describe this construction in Section 3.4.1, and bound its condition number in Section 3.4.2.

3.4.1 Construction

In this section, we show the construction from an instance of $\mathcal{MC}_2$ to an instance of $\mathcal{MC}_2^{>0}$. The strategy is to add extra edges with a sufficiently small weight, such that $\boldsymbol{L}^1, \boldsymbol{L}^2, \boldsymbol{L}^{1+2}$ have identical nonzero stricture and the solution of the linear system does not change much.

The reduction from $\mathcal{MC}_2$ to $\mathcal{MC}_2^{>0}$, with pseudo-code in Algorithm 4, simply adds edges with weight δ to all the missing edges. The transformation of solutions of the corresponding instance of $\mathcal{MC}_2^{>0}$ back to a solution of the original $\mathcal{MC}_2$ instance in Algorithm 5 simply returns the same vector $\boldsymbol{x}$.

Algorithm 4 REDUCE$\mathcal{MC}_2$TO$\mathcal{MC}_2^{>0}$

Input: $(\boldsymbol{B}, \boldsymbol{c}^{\mathrm{B}}, \epsilon_1)$ where $\boldsymbol{B} \in \mathcal{MC}_2$ is an $m \times n$ matrix, $\boldsymbol{c}^{\mathrm{B}} \in \mathbb{R}^m$, and $0 < \epsilon_1 < 1$.

Output: $(\boldsymbol{B}^{>0}, \boldsymbol{c}^{\mathrm{B}^{>0}}, \epsilon_2)$ where $\boldsymbol{B}^{>0} \in \mathcal{MC}_2^{>0}$ is an $m' \times n'$ matrix, $\boldsymbol{c}^{\mathrm{B}^{>0}} \in \mathbb{R}^{m'}$, and $0 < \epsilon_2 < 1$.

$\delta \leftarrow \frac{\epsilon_1}{100\kappa(\boldsymbol{B})\sigma_{\max}(\boldsymbol{B})\|\boldsymbol{c}^{\mathrm{B}}\|_2}$

$\widehat{\mathcal{B}} \leftarrow \emptyset$ {New equations for $\mathcal{MC}_2^{>0}$}

for each pair of vertices i, j whose blocks are involved in some equation in $\boldsymbol{B}$ **do**

  if $\boldsymbol{B}$ does not contain a type 1 equation between i and j **then**

    $\widehat{\mathcal{B}} \leftarrow \widehat{\mathcal{B}} \cup \{\boldsymbol{u}_i - \boldsymbol{u}_j = 0\}$.

  end if

  if $\boldsymbol{B}$ does not contain a type 2 equation betwen i and j **then**

    $\widehat{\mathcal{B}} \leftarrow \widehat{\mathcal{B}} \cup \{\boldsymbol{v}_i - \boldsymbol{v}_j = 0\}$.

  end if

  if $\boldsymbol{B}$ does not contain a type $1+2$ equation betwen i and j **then**

    $\widehat{\mathcal{B}} \leftarrow \widehat{\mathcal{B}} \cup \{\boldsymbol{u}_i + \boldsymbol{v}_i - (\boldsymbol{u}_j + \boldsymbol{v}_j) = 0\}$.

  end if

end for

Let $\widetilde{\boldsymbol{B}}$ be the coefficient matrix of equations in $\widehat{\mathcal{B}}$.

return

$$\boldsymbol{B}^{>0} = \begin{pmatrix} \boldsymbol{B} \\ \delta\widetilde{\boldsymbol{B}} \end{pmatrix}, \qquad \boldsymbol{c}^{\mathrm{B}^{>0}} = \begin{pmatrix} \boldsymbol{c}^{\mathrm{B}} \\ \boldsymbol{0} \end{pmatrix}, \qquad \epsilon_2 = \frac{\epsilon_1}{100}.$$

Algorithm 5 MAPSOLN$\mathcal{MC}_2^{>0}$TO$\mathcal{MC}_2$

Input: $m \times n$ matrix $\boldsymbol{B} \in \mathcal{MC}_2$, $m' \times n'$ matrix $\boldsymbol{B}^{>0} \in \mathcal{MC}_2^{>0}$, vector $\boldsymbol{c}^{\mathrm{B}} \in \mathbb{R}^m$ vector $\boldsymbol{x} \in \mathbb{R}^n$.

Output: Vector $\boldsymbol{y} \in \mathbb{R}^n$.

if $\boldsymbol{B}^\top \boldsymbol{c}^{\mathrm{B}} = \boldsymbol{0}$ **then**

  return $\boldsymbol{y} \leftarrow \boldsymbol{0}$

else

  return $\boldsymbol{y} \leftarrow \boldsymbol{x}$

end if

Let $\boldsymbol{B}^{>0}\boldsymbol{x} = \boldsymbol{c}^{\mathrm{B}^{>0}}$ be the linear system returned by a call to REDUCE$\mathcal{MC}_2$TO$\mathcal{MC}_2^{>0}(\boldsymbol{B}, \boldsymbol{c}^{\mathrm{B}}, \epsilon_1)$. As we only add up to two edges per original edge in $\boldsymbol{B}$, and do not introduce any new variables, the size of $\boldsymbol{B}^{>0}$ is immediate from this routine. According to the algorithms, $\boldsymbol{c}^{\mathrm{B}^{>0}}$ is simply

$$\boldsymbol{c}^{\mathrm{B}^{>0}} = \begin{pmatrix} \boldsymbol{c}^{\mathrm{B}} \\ \boldsymbol{0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{c}^{\mathrm{A}} \\ \boldsymbol{0} \end{pmatrix}.$$

Note the two zero vectors in the above equation have different dimensions.

For a given matrix $\boldsymbol{B}$, we will choose the additive term to ensure non-zeros to be:

$$\delta \stackrel{\text{def}}{=} \frac{\epsilon_1}{100\kappa(\boldsymbol{B})\sigma_{\max}(\boldsymbol{B}) \left\| \boldsymbol{c}^{\mathrm{B}} \right\|_2}. \tag{3.21}$$

Note that we can bound $\kappa(\boldsymbol{B}), \sigma_{\max}(\boldsymbol{B})$ by condition number of $\boldsymbol{A}$ (i.e, the linear system instance of $\mathcal{G}$).

Lemma 3.4.1. $\mathrm{nnz}(\boldsymbol{B}^{>0}) = O\left(\mathrm{nnz}(\boldsymbol{B})\right)$. *By our setting of δ in Equation* (3.21)*, the largest entry of $\boldsymbol{B}^{>0}$ does not change, the smallest entry of $\boldsymbol{B}^{>0}$ is at least δ.*

Note that the addition of all three types of edges means that the null space of $\boldsymbol{B}^{>0}$ is now given by the connected components in its graph theoretic structure, and is likely significantly different from the null space of $\boldsymbol{B}$. In order to solve the Linear System Approximation Problem(LSA) problems

$$\min_{\boldsymbol{x}} \left\| \boldsymbol{B}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{B}} \right\|_2,$$
$$\min_{\boldsymbol{x}} \left\| \boldsymbol{B}^{>0}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{B}^{>0}} \right\|_2,$$

we need to solve the two linear systems:

$$\boldsymbol{B}^\top \boldsymbol{B}\boldsymbol{x} = \boldsymbol{B}^\top \boldsymbol{c}^{\mathrm{B}}, \tag{3.22}$$
$$(\boldsymbol{B}^{>0})^\top \boldsymbol{B}^{>0}\boldsymbol{x} = (\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}}. \tag{3.23}$$

First note that the RHS the two equations are the same because:

$$(\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}} = \begin{pmatrix} \boldsymbol{B}^\top & \delta\widetilde{\boldsymbol{B}}^\top \end{pmatrix} \begin{pmatrix} \boldsymbol{c}^{\mathrm{B}} \\ \mathbf{0} \end{pmatrix} = \boldsymbol{B}^\top \boldsymbol{c}^{\mathrm{B}}.$$

This means that for a sufficiently small choice of δ, the differences between the solu-

tions of these two linear systems is small.

Lemma 3.4.2. *Let $\boldsymbol{B}^{>0}\boldsymbol{x} = \boldsymbol{c}^{B^{>0}}$ be the linear system returned by a call to* REDUCE$\mathcal{MC}_2$TO$\mathcal{MC}_2^{>0}(\boldsymbol{B}, \boldsymbol{c}^B, \epsilon_1)$. *Let $\boldsymbol{x}^{B*} \in \arg\min_{\boldsymbol{x}} \|\boldsymbol{B}\boldsymbol{x} - \boldsymbol{c}^B\|_2$ and $\boldsymbol{x}^{B*} \perp \text{null}(\boldsymbol{B})$. Let $\boldsymbol{x}^{B^{>0}*} \in \arg\min_{\boldsymbol{x}} \left\|\boldsymbol{B}^{>0}\boldsymbol{x} - \boldsymbol{c}^{B^{>0}}\right\|_2$. Then we have:*

$$\left\|\boldsymbol{x}^{B*} - \boldsymbol{x}^{B^{>0}*}\right\|_{\boldsymbol{B}^\top\boldsymbol{B}} \leq O\left(\delta(1+\delta)\kappa(\boldsymbol{B})\left\|\boldsymbol{c}^B\right\|_2\right).$$

Proof. The desired distance can be written as

$$\left\|\boldsymbol{x}^{\mathrm{B}*} - \boldsymbol{x}^{\mathrm{B}^{>0}*}\right\|_{\boldsymbol{B}^\top\boldsymbol{B}} = \left\|\boldsymbol{B}\left(\boldsymbol{x}^{\mathrm{B}*} - \boldsymbol{x}^{\mathrm{B}^{>0}*}\right)\right\|_2.$$

Also, the optimality of $\boldsymbol{x}^{\mathrm{B}*}$ means that $\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}*} - \boldsymbol{c}^{\mathrm{B}}$ is perpendicular to anything in the rank space of $\boldsymbol{B}$, in particular,

$$\boldsymbol{B}\left(\boldsymbol{x}^{\mathrm{B}*} - \boldsymbol{x}^{\mathrm{B}^{>0}*}\right) \perp \boldsymbol{B}\boldsymbol{x}^{\mathrm{B}*} - \boldsymbol{c}^{\mathrm{B}},$$

which in turn gives:

$$\left\|\boldsymbol{B}\left(\boldsymbol{x}^{\mathrm{B}*} - \boldsymbol{x}^{\mathrm{B}^{>0}*}\right)\right\|_2^2 = \left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}^{>0}*} - \boldsymbol{c}^{\mathrm{B}}\right\|_2^2 - \left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}*} - \boldsymbol{c}^{\mathrm{B}}\right\|_2^2. \tag{3.24}$$

We now bound the right hand side. Since $\boldsymbol{x}^{\mathrm{B}^{>0}*}$ is a minimizer of $\left\|\boldsymbol{B}^{>0}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2$, we have

$$\left\|\boldsymbol{B}^{>0}\boldsymbol{x}^{\mathrm{B}^{>0}*} - \boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2^2 \leq \left\|\boldsymbol{B}^{>0}\boldsymbol{x}^{\mathrm{B}*} - \boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2^2,$$

where we can extract out the $\delta\widetilde{\boldsymbol{B}}$ term in $\boldsymbol{B}^{>0}$ separately to get:

$$\left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}^{>0}*} - \boldsymbol{c}^{\mathrm{B}}\right\|_2^2 + \delta^2\left\|\widetilde{\boldsymbol{B}}\boldsymbol{x}^{\mathrm{B}^{>0}*}\right\|_2^2 \leq \left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}*} - \boldsymbol{c}^{\mathrm{B}}\right\|_2^2 + \delta^2\left\|\widetilde{\boldsymbol{B}}\boldsymbol{x}^{\mathrm{B}*}\right\|_2^2.$$

Together with Equation (3.24), this then gives

$$\left\|\boldsymbol{x}^{\mathrm{B}*}-\boldsymbol{x}^{\mathrm{B}^{>0}*}\right\|_{\boldsymbol{B}^\top\boldsymbol{B}} \leq \delta\left\|\widetilde{\boldsymbol{B}}\boldsymbol{x}^{\mathrm{B}*}\right\|_2.$$

It remains to upper bound $\left\|\widetilde{\boldsymbol{B}}\boldsymbol{x}^{\mathrm{B}^{>0}*}\right\|_2$. By the assumption of $\boldsymbol{x}^{\mathrm{B}*} \perp \mathrm{null}(\boldsymbol{B})$, we get:

$$\frac{\left\|\widetilde{\boldsymbol{B}}\boldsymbol{x}^{\mathrm{B}*}\right\|_2^2}{\left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}*}\right\|_2^2} \leq \frac{\lambda_{\max}\left(\widetilde{\boldsymbol{B}}^\top\widetilde{\boldsymbol{B}}\right)}{\lambda_{\min}\left(\boldsymbol{B}^\top\boldsymbol{B}\right)}.$$

By a proof similar to Lemma 3.3.8, which upper bounds $\lambda_{\max}\left(\boldsymbol{B}^\top\boldsymbol{B}\right)$, we have:

$$\sigma_{\max}\left(\widetilde{\boldsymbol{B}}\right) = O(1+\delta)\sigma_{\max}(\boldsymbol{B}),$$

which implies

$$\left\|\widetilde{\boldsymbol{B}}\boldsymbol{x}^{\mathrm{B}*}\right\|_2 \leq O(1+\delta)\kappa(\boldsymbol{B})\left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}*}\right\|_2.$$

Since $\boldsymbol{x}^{\mathrm{B}*}$ is a minimizer of $\min_{\boldsymbol{x}}\left\|\boldsymbol{B}\boldsymbol{x}-\boldsymbol{c}^{\mathrm{B}}\right\|_2$, we have

$$\left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}*}\right\|_2 = \left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2 \leq \left\|\boldsymbol{c}^{\mathrm{B}}\right\|_2.$$

Therefore,

$$\left\|\boldsymbol{x}^{\mathrm{B}*}-\boldsymbol{x}^{\mathrm{B}^{>0}*}\right\|_{\boldsymbol{B}^\top\boldsymbol{B}} \leq O\left(\delta(1+\delta)\kappa(\boldsymbol{B})\right)\|\boldsymbol{c}\|_2,$$

which completes the proof. □

We now check that the approximate solutions of the two linear systems in Equation (3.22) and (3.23) are also close to each other.

Lemma 3.4.3. *Let $\boldsymbol{B}^{>0}\boldsymbol{x} = \boldsymbol{c}^{B^{>0}}$ be the linear system returned by a call to* REDUCE$\mathcal{MC}_2$TO$\mathcal{MC}_2^{>0}(\boldsymbol{B}, \boldsymbol{c}^B, \epsilon_1)$*, and let ϵ_2 be the error parameter returned by this call.*

Let $\boldsymbol{x}$ be a vector such that

$$\left\|\boldsymbol{B}^{>0}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\boldsymbol{B}^{>0}}\right\|_2 \leq \epsilon_2 \left\|\boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\boldsymbol{B}^{>0}}\right\|_2.$$

Let $\boldsymbol{y}$ be the vector returned by a call to MAPSOLN$\mathcal{MC}_2^{>0}$TO$\mathcal{MC}_2(\boldsymbol{B}, \boldsymbol{B}^{>0}, \boldsymbol{c}^{\boldsymbol{B}}, \boldsymbol{x})$*. Then*

$$\left\|\boldsymbol{B}\boldsymbol{y} - \boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\boldsymbol{B}}\right\|_2 \leq \epsilon_1 \left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\boldsymbol{B}}\right\|_2.$$

Proof. In Algorithm 5, if the condition $\boldsymbol{B}^\top \boldsymbol{c}^{\mathrm{B}} = \boldsymbol{0}$ is true, then $\boldsymbol{y} = \boldsymbol{0}$. This implies

$$\left\|\boldsymbol{B}\boldsymbol{y} - \boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2 = \boldsymbol{0}.$$

In the following, we assume that $\boldsymbol{B}^\top \boldsymbol{c}^{\mathrm{B}} \neq \boldsymbol{0}$, in which case $\boldsymbol{y} = \boldsymbol{x}$. We first show that the construction implies $\left\|\boldsymbol{B}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2 \leq \left\|\boldsymbol{B}^{>0}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2$. Once again, let $\boldsymbol{x}^{\mathrm{B}*}$ and $\boldsymbol{x}^{\mathrm{B}^{>0}*}$ be the vectors that minimize $\left\|\boldsymbol{B}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{B}}\right\|_2$ and $\left\|\boldsymbol{B}^{>0}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2$ respectively. This choice gives:

$$\left\|\boldsymbol{B}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2 = \left\|\boldsymbol{x} - \boldsymbol{x}^{\mathrm{B}*}\right\|_{\boldsymbol{B}^\top\boldsymbol{B}}.$$

As $\|\cdot\|_{\boldsymbol{B}^\top\boldsymbol{B}}$ is a norm, by triangle inequality we have:

$$\left\|\boldsymbol{x} - \boldsymbol{x}^{\mathrm{B}*}\right\|_{\boldsymbol{B}^\top\boldsymbol{B}} \leq \left\|\boldsymbol{x} - \boldsymbol{x}^{\mathrm{B}^{>0}*}\right\|_{\boldsymbol{B}^\top\boldsymbol{B}} + \left\|\boldsymbol{x}^{\mathrm{B}^{>0}*} - \boldsymbol{x}^{\mathrm{B}*}\right\|_{\boldsymbol{B}^\top\boldsymbol{B}}.$$

We will bound these two terms separately.

Since $\boldsymbol{B}^\top\boldsymbol{B} \preccurlyeq \boldsymbol{B}^\top\boldsymbol{B} + \delta^2\widetilde{\boldsymbol{B}}^\top\widetilde{\boldsymbol{B}} = (\boldsymbol{B}^{>0})^\top\boldsymbol{B}^{>0}$, the first term is less than its norm in the $(\boldsymbol{B}^{>0})^\top\boldsymbol{B}^{>0}$ norm:

$$\left\|\boldsymbol{x} - \boldsymbol{x}^{\mathrm{B}^{>0}*}\right\|_{\boldsymbol{B}^\top\boldsymbol{B}} \leq \left\|\boldsymbol{x} - \boldsymbol{x}^{\mathrm{B}^{>0}*}\right\|_{\left(\boldsymbol{B}^\top\boldsymbol{B}+\delta^2\widetilde{\boldsymbol{B}}^\top\widetilde{\boldsymbol{B}}\right)} = \left\|\boldsymbol{B}^{>0}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2,$$

while the second term is precisely the distances between the two optimums, which we just

bounded in Lemma 3.4.2. Combining these bounds then gives:

$$\left\|\boldsymbol{B}\boldsymbol{x}-\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2 \leq \left\|\boldsymbol{B}^{>0}\boldsymbol{x}-\boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2 + O\left(\delta(1+\delta)\kappa(\boldsymbol{B})\left\|\boldsymbol{c}^{\mathrm{B}}\right\|_2\right). \tag{3.25}$$

As the equations in $\boldsymbol{B}^{>0}$ is a superset of the ones in $\boldsymbol{B}$, we have

$$\left\|\boldsymbol{B}^{>0}\boldsymbol{x}^{\mathrm{B}^{>0}*}-\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2 \geq \left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}^{>0}*}-\boldsymbol{c}^{\mathrm{B}}\right\|_2 \geq \left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}*}-\boldsymbol{c}^{\mathrm{B}}\right\|_2.$$

Substituting $\left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}*}-\boldsymbol{c}^{\mathrm{B}}\right\|_2^2 = \left\|\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 - \left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2^2$ and its equivalent in $\boldsymbol{B}^{>0}$ gives

$$\left\|\boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2 \leq \left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2.$$

Together with the condition of the lemma,

$$\left\|\boldsymbol{B}^{>0}\boldsymbol{x}-\boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2 \leq \epsilon_2\left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2.$$

Plugging this into Equation (3.25), we have

$$\left\|\boldsymbol{B}\boldsymbol{x}-\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2 = O\left(\epsilon_2\left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2 + \delta\kappa\left(\boldsymbol{B}\right)\left\|\boldsymbol{c}^{\mathrm{B}}\right\|_2\right).$$

It remains to upper bound $\left\|\boldsymbol{c}^{\mathrm{B}}\right\|_2$ by a function of $\left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2$. By our construction in Algorithm 1 and assumption, $\left\|\boldsymbol{B}\boldsymbol{c}^{\mathrm{B}}\right\|_2^2$ is a positive integer. By Lemma 2.2.5,

$$\left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2 \geq \frac{1}{\sigma_{\max}(\boldsymbol{B})}.$$

Thus,

$$\left\|\boldsymbol{B}\boldsymbol{x}-\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2 = O\left(\epsilon_2+\delta\kappa(\boldsymbol{B})\sigma_{\max}(\boldsymbol{B})\left\|\boldsymbol{c}^{\mathrm{B}}\right\|_2\right)\left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2.$$

By our setting of δ in Equation (3.21), we have

$$\left\|\boldsymbol{B}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2 \leq \epsilon_1 \left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2.$$

This completes the proof. □

3.4.2 Bounding Condition Number of the New System in $\mathcal{MC}_2^{>0}$

We now establish bounds on the numerical quantities related to $\boldsymbol{B}^{>0}$. By a proof similar to the upper bound on $\lambda_{\max}\left(\boldsymbol{B}^\top\boldsymbol{B}\right)$ in Lemma 3.3.8, we have:

$$\lambda_{\max}\left((\boldsymbol{B}^{>0})^\top\boldsymbol{B}^{>0}\right) = O\left(\lambda_{\max}\left(\boldsymbol{B}^\top\boldsymbol{B}\right)\right).$$

As a result, we focus on the lower bound here:

Lemma 3.4.4. *The matrix $\boldsymbol{B}^{>0}$ from the linear system returned by a call to* REDUCE$\mathcal{MC}_2$TO$\mathcal{MC}_2^{>0}(\boldsymbol{B}, \boldsymbol{c}^B, \epsilon_1, \delta)$*, where δ is set according to Equation 3.21, satisfies*

$$\sigma_{\min}\left(\boldsymbol{B}^{>0}\right) = \Omega\left(\frac{\epsilon_1}{\kappa(\boldsymbol{B})\sigma_{\max}(\boldsymbol{B})n_B}\right).$$

Proof. Let G be a unit-edge weight graph whose vertex set and edge set are same as the underlying graph of $\boldsymbol{B}^{>0}$. Let $\boldsymbol{L}_G$ be the associated Laplacian matrix of G, and let $\boldsymbol{M} := \boldsymbol{L}_G \otimes \boldsymbol{C}$, where

$$\boldsymbol{C} = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

is a symmetric PSD matrix. Note

$$\boldsymbol{M} = \boldsymbol{L}_G \otimes \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \boldsymbol{L}_G \otimes \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} + \boldsymbol{L}_G \otimes \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},$$

which means $\boldsymbol{M}$ has the same null space as $(\boldsymbol{B}^{>0})^\top \boldsymbol{B}^{>0}$.

$$\lambda_{\min}\left((\boldsymbol{B}^{>0})^\top \boldsymbol{B}^{>0}\right) = \min_{\boldsymbol{x}\perp \text{null}(\boldsymbol{B}^{>0})} \frac{\boldsymbol{x}^\top (\boldsymbol{B}^{>0})^\top \boldsymbol{B}^{>0}\boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}} \geq \delta^2 \min_{\boldsymbol{x}\perp \text{null}(\boldsymbol{M})} \frac{\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}},$$

the last inequality is due to that δ is the minimum edge weight.

Note that the sum of the 3 types of blocks is positive definite and has eigenvalue at least 1: the type 1 and 2 blocks already sum to $\boldsymbol{I}$. Formally:

$$\lambda_{\min}(\boldsymbol{M}) = \lambda_{\min}(\boldsymbol{L}_G) \cdot \lambda_{\min}(\boldsymbol{C}) = \lambda_{\min}(\boldsymbol{L}_G).$$

The result then follows from the folklore bound that the minimum non-zero eigenvalue of a unit weighted graph is at least $\frac{1}{n^2}$. One way to see this is via Cheeger's inequality (see e.g. [69] applied to each block: this decomposition is equivalent to spearating the matrix into its diagonal blocks based on the connected components, and then invoking the fact that the minimum weight of a cut is at least 1, and there are at most n vertices. Together these tools imply $\lambda_{\min}(\boldsymbol{L}_G) = \Omega\left(\frac{1}{n_B^2}\right)$, and in turn:

$$\sigma_{\min}\left(\boldsymbol{B}^{>0}\right) \geq \Omega\left(\frac{\delta}{n}\right) = \Omega\left(\frac{\epsilon_1}{\kappa(\boldsymbol{B})\sigma_{\max}(\boldsymbol{B})n_B}\right).$$

□

This also implies a bound on the condition number of $\boldsymbol{B}^{>0}$:

Lemma 3.4.5. $\kappa(\boldsymbol{B}^{>0}) = O\left(\epsilon_1^{-1}\sigma_{\max}^2(\boldsymbol{B})\kappa(\boldsymbol{B})n_B\right)$.

3.4.3 Putting it All Together

Proof of Lemma 3.2.6. By Lemma 3.4.1, we have

$$\text{nnz}(\boldsymbol{B}^{>0}) = O(s),$$

The largest entry is U, and the smallest nonzero entry is δ. By Equation (3.21) and Lemma 3.1.2 and 3.1.2,

$$\delta = \Omega\left(\frac{\epsilon}{KU^2}\right).$$

By Lemma 3.4.5 and Lemma 3.1.2 and 3.1.2, the condition number is

$$\kappa(\boldsymbol{B}^{>0}) = O\left(\frac{s^2U^2K}{\epsilon}\right).$$

By Lemma 3.4.3, the accuracy error for $\boldsymbol{B}^{>0}$ is $O(\epsilon)$. □

3.5 Rounding and Scaling Weights to Integers

In this section, we show a reduction from the linear system with strict 2-commodity matrix to the linear system with integral strict 2-commodity matrix.

Algorithm 6 presents the pseudo-code for the algorithm REDUCE$\mathcal{MC}_2^{>0}$TO$\mathcal{MC}_{2,\mathbb{Z}}^{>0}$. Given an instance $(\boldsymbol{B}^{>0}, \boldsymbol{c}^{\mathrm{B}^{>0}}, \epsilon_1)$ where $\boldsymbol{B}^{>0} \in \mathcal{MC}_2^{>0}$, the class of strict 2-commodity matrices, the call REDUCE$\mathcal{MC}_2^{>0}$TO$\mathcal{MC}_{2,\mathbb{Z}}^{>0}(\boldsymbol{B}^{>0}, \boldsymbol{c}^{\mathrm{B}^{>0}}, \epsilon_1)$ returns an instance $(\boldsymbol{B}^{>0,\mathbb{Z}}, \boldsymbol{c}^{\boldsymbol{B}^{>0,\mathbb{Z}}}, \epsilon_2)$ where $\boldsymbol{B}^{>0,\mathbb{Z}} \in \mathcal{MC}_{2,\mathbb{Z}}^{>0}$. Algorithm 7 provides the (trivial) pseudo-code for MAPSOLN$\mathcal{MC}_{2,\mathbb{Z}}^{>0}TO\mathcal{MC}_2^{>0}$ which maps a solution of an instance over $\mathcal{MC}_{2,\mathbb{Z}}^{>0}$ to a solution of an instance over $\mathcal{MC}_2^{>0}$.

For simplicity, we analyze an intermediate linear system $\boldsymbol{B}^{int}\boldsymbol{y} = \boldsymbol{c}^{\mathrm{B}^{>0}}$, where

$$(\boldsymbol{B}^{int})_{ij} \stackrel{\text{def}}{=} 2^{-k} \cdot (\boldsymbol{B}^{>0,\mathbb{Z}})_{ij}.$$

Recall the definition of $\boldsymbol{B}^{>0,\mathbb{Z}}$ in Algorithm 6 line 5, we have

$$(\boldsymbol{B}^{int})_{ij} = 2^{-k} \cdot \left\lceil (\boldsymbol{B}^{>0})_{ij} \cdot 2^k \right\rceil. \tag{3.26}$$

Algorithm 6 REDUCE$\mathcal{MC}_2^{>0}$TO$\mathcal{MC}_{2,\mathbb{Z}}^{>0}$

Input: $(\boldsymbol{B}^{>0}, \boldsymbol{c}^{\mathrm{B}^{>0}}, \epsilon_1)$ where $\boldsymbol{B}^{>0} \in \mathcal{MC}_2^{>0}$ is an $m \times n$ matrix, $\boldsymbol{c}^{\mathrm{B}^{>0}} \in \mathbb{R}^m$, and $0 < \epsilon_1 < 1$

Output: $(\boldsymbol{B}^{>0,\mathbb{Z}}, \boldsymbol{c}^{\boldsymbol{B}^{>0,\mathbb{Z}}}, \epsilon_2)$ where $\boldsymbol{B}^{>0,\mathbb{Z}} \in \mathcal{MC}_{2,\mathbb{Z}}^{>0}$ is an $m' \times n'$ matrix, $\boldsymbol{c}^{\boldsymbol{B}^{>0,\mathbb{Z}}} \in \mathbb{R}^{m'}$, and $0 < \epsilon_2 < 1$.

$k \leftarrow \left\lceil \log_2 \frac{\mathrm{nnz}(\boldsymbol{B}^{>0})}{\epsilon_1} \right\rceil + 2$
$\boldsymbol{B}^{>0,\mathbb{Z}} \leftarrow \boldsymbol{B}^{>0}$
$\boldsymbol{c}^{\boldsymbol{B}^{>0,\mathbb{Z}}} \leftarrow \boldsymbol{c}^{\mathrm{B}^{>0}}$
for each entry $(\boldsymbol{B}^{>0,\mathbb{Z}})_{ij}$ **do**
  $(\boldsymbol{B}^{>0,\mathbb{Z}})_{ij} \leftarrow \left\lceil (\boldsymbol{B}^{>0,\mathbb{Z}})_{ij} \cdot 2^k \right\rceil$
end for
for each entry $\boldsymbol{c}_i^{\boldsymbol{B}^{>0,\mathbb{Z}}}$ **do**
  $\boldsymbol{c}_i^{\boldsymbol{B}^{>0,\mathbb{Z}}} \leftarrow \boldsymbol{c}_i^{\boldsymbol{B}^{>0,\mathbb{Z}}} \cdot 2^k$
end for
return $(\boldsymbol{B}^{>0,\mathbb{Z}}, \boldsymbol{c}^{\boldsymbol{B}^{>0,\mathbb{Z}}}, \epsilon_1/3)$.

Algorithm 7 MAPSOLN$\mathcal{MC}_{2,\mathbb{Z}}^{>0}TO\mathcal{MC}_2^{>0}$

Input: $m \times n$ matrix $\boldsymbol{B}^{>0} \in \mathcal{MC}_2$, $m' \times n'$ matrix $\boldsymbol{B}^{>0,\mathbb{Z}} \in \mathcal{MC}_2^{>0}$, vector $\boldsymbol{x} \in \mathbb{R}^{n'}$.

Output: Vector $\boldsymbol{y} \in \mathbb{R}^n$.

return $\boldsymbol{x}$

The linear system $\boldsymbol{B}^{int}\boldsymbol{y} = \boldsymbol{c}^{\mathrm{B}^{>0}}$ is exactly the linear system $\boldsymbol{B}^{>0,\mathbb{Z}}\boldsymbol{y} = \boldsymbol{c}^{\boldsymbol{B}^{>0,\mathbb{Z}}}$ multiplying a factor 2^{-k} on both sides.

Note the condition number and the eigen-space of a matrix, the optimal solutions of the projection problems are invariant under scaling. Let $\boldsymbol{y}^* \in \arg\min_{\boldsymbol{y}} \left\| \boldsymbol{B}^{>0,\mathbb{Z}}\boldsymbol{y} - \boldsymbol{c}^{\boldsymbol{B}^{>0,\mathbb{Z}}} \right\|_2 = \arg\min_{\boldsymbol{y}} \left\| \boldsymbol{B}^{int}\boldsymbol{y} - \boldsymbol{c}^{\mathrm{B}^{>0}} \right\|_2$. The following two inequalities:

$$\|\boldsymbol{y} - \boldsymbol{y}^*\|_{(\boldsymbol{B}^{>0,\mathbb{Z}})^\top \boldsymbol{B}^{>0,\mathbb{Z}}} \leq \epsilon \left\| \boldsymbol{\Pi}_{\boldsymbol{B}^{>0,\mathbb{Z}}} \boldsymbol{c}^{\boldsymbol{B}^{>0,\mathbb{Z}}} \right\|_2$$

and

$$\|\boldsymbol{y} - \boldsymbol{y}^*\|_{(\boldsymbol{B}^{int})^\top \boldsymbol{B}^{int}} \leq \epsilon \left\| \boldsymbol{\Pi}_{\boldsymbol{B}^{int}} \boldsymbol{c}^{\mathrm{B}^{>0}} \right\|_2$$

are equivalent. Thus, it suffices to analyze the linear system $\boldsymbol{B}^{int}\boldsymbol{y} = \boldsymbol{c}^{\mathrm{B}^{>0}}$.

Solving the two projection problems

$$\min_{\boldsymbol{x}} \left\| \boldsymbol{B}^{>0}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{B}^{>0}} \right\|_2,$$
$$\min_{\boldsymbol{y}} \left\| \boldsymbol{B}^{int}\boldsymbol{y} - \boldsymbol{c}^{\mathrm{B}^{>0}} \right\|_2,$$

is equivalent to solving the following two linear systems:

$$(\boldsymbol{B}^{>0})^\top \boldsymbol{B}^{>0}\boldsymbol{x} = (\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}}, \tag{3.27}$$
$$(\boldsymbol{B}^{int})^\top \boldsymbol{B}^{int}\boldsymbol{y} = (\boldsymbol{B}^{int})^\top \boldsymbol{c}^{\mathrm{B}^{>0}}.$$

Note $\boldsymbol{c}^{\mathrm{B}^{>0}} = (\boldsymbol{c}^{\mathrm{A}}; \boldsymbol{0})$, and all entries of the rows of $\boldsymbol{B}^{>0}$ corresponding to the original rows of $\boldsymbol{A}$ are integers. Thus,

$$(\boldsymbol{B}^{int})^\top \boldsymbol{c}^{\mathrm{B}^{>0}} = (\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}}.$$

That is, the 2nd linear system is equivalent to

$$(\boldsymbol{B}^{int})^\top \boldsymbol{B}^{int}\boldsymbol{y} = (\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}}. \tag{3.28}$$

Let

$$\boldsymbol{M} \stackrel{\text{def}}{=} (\boldsymbol{B}^{>0})^\top \boldsymbol{B}^{>0} \text{ and } \widehat{\boldsymbol{M}} \stackrel{\text{def}}{=} (\boldsymbol{B}^{int})^\top \boldsymbol{B}^{int}.$$

We bound the eigenvalues of $\widehat{\boldsymbol{M}}$ by the following lemma.

Lemma 3.5.1. *$\lambda_{\max}(\widehat{\boldsymbol{M}}) \leq (1 + 2^{-k+2})\lambda_{\max}(\boldsymbol{M})$ and $\lambda_{\min}(\widehat{\boldsymbol{M}}) \geq \lambda_{\min}(\boldsymbol{M})$. Furthermore, $\kappa(\widehat{\boldsymbol{M}}) \leq (1 + 2^{-k+2})\kappa(\boldsymbol{M})$.*

Proof. Note $\boldsymbol{M}$ can be written as $\boldsymbol{B}^\top \boldsymbol{W}\boldsymbol{B}$, where $\boldsymbol{B}$ is the incidence-structured block matrix with *unit* nonzero entries and $\boldsymbol{W}$ is the diagonal matrix with edge weights. Let $\boldsymbol{x}$

be an arbitrary vector, and let $\boldsymbol{y} \stackrel{\text{def}}{=} \boldsymbol{B}\boldsymbol{x}$.

$$\boldsymbol{x}^\top \boldsymbol{M} \boldsymbol{x} = \boldsymbol{y}^\top \boldsymbol{W} \boldsymbol{y} = \sum_i \boldsymbol{W}_{ii} \boldsymbol{y}_i^2,$$

and

$$\boldsymbol{x}^\top \widehat{\boldsymbol{M}} \boldsymbol{x} = \sum_i \widehat{\boldsymbol{W}}_{i} \boldsymbol{y}_i^2.$$

Let

$$\alpha \stackrel{\text{def}}{=} \min_i \frac{\widehat{\boldsymbol{W}}_{ii}}{\boldsymbol{W}_{ii}} \text{ and } \beta \stackrel{\text{def}}{=} \max_i \frac{\widehat{\boldsymbol{W}}_{ii}}{\boldsymbol{W}_{ii}}.$$

We have

$$\alpha \sum_i \boldsymbol{W}_{i} \boldsymbol{y}_i^2 \leq \sum_i \widehat{\boldsymbol{W}}_{i} \boldsymbol{y}_i^2 \leq \beta \sum_i \boldsymbol{W}_{i} \boldsymbol{y}_i^2.$$

This implies that

$$\alpha \boldsymbol{M} \preccurlyeq \widehat{\boldsymbol{M}} \preccurlyeq \beta \boldsymbol{M}. \tag{3.29}$$

Now we bound the values of α, β. According to Equation (3.26),

$$\alpha = \min_i \frac{\widehat{\boldsymbol{W}}_{ii}}{\boldsymbol{W}_{ii}} \geq \frac{\left(2^{-k} \cdot \boldsymbol{W}_{ii}^{1/2} \cdot 2^k\right)^2}{\boldsymbol{W}_{ii}} = 1. \tag{3.30}$$

Similarly,

$$\beta = \max_i \frac{\widehat{\boldsymbol{W}}_{ii}}{\boldsymbol{W}_{ii}} \leq \frac{\left(2^{-k}\left(\boldsymbol{W}_{ii}^{1/2} \cdot 2^k + 1\right)\right)^2}{\boldsymbol{W}_{ii}} \leq 1 + 2^{-k+2}. \tag{3.31}$$

The last inequality is due to $\boldsymbol{W}_{ii} \geq 1$. Thus,

$$\lambda_{\max}(\widehat{\boldsymbol{M}}) \leq (1 + 2^{-k+2}) \lambda_{\max}(\boldsymbol{M}) \text{ and } \lambda_{\min}(\widehat{\boldsymbol{M}}) \geq \lambda_{\min}(\boldsymbol{M}).$$

The bound on the condition number $\kappa(\widehat{\boldsymbol{M}}) \leq (1 + 2^{-k+2}) \kappa(\boldsymbol{M})$ immediately follows the

above two inequalities. □

We show the exact solutions of the two linear systems in Equation (3.27) and (3.28) are close, by the following lemma.

Lemma 3.5.2. *Let*

$$\boldsymbol{x}^* \stackrel{\text{def}}{=} \boldsymbol{M}^\dagger (\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{B^{>0}} \text{ and } \boldsymbol{y}^* \stackrel{\text{def}}{=} \widehat{\boldsymbol{M}}^\dagger (\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{B^{>0}}.$$

$\boldsymbol{x}^*$ *and* $\boldsymbol{y}^*$ *are exact solutions of linear system* (3.27) *and* (3.28) *respectively.*

$$\|\boldsymbol{y}^* - \boldsymbol{x}^*\|_{\boldsymbol{M}} \leq 2^{-k+2} \left\| (\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{B^{>0}} \right\|_{\boldsymbol{M}^\dagger}.$$

Proof. Note $\boldsymbol{M}, \widehat{\boldsymbol{M}}$ are both symmetric. Expanding the left hand side,

$$\begin{aligned}
\|\boldsymbol{y}^* - \boldsymbol{x}^*\|_2^2 &= (\boldsymbol{c}^{\mathrm{B}^{>0}})^\top \boldsymbol{B}^{>0} \left(\widehat{\boldsymbol{M}}^\dagger - \boldsymbol{M}^\dagger \right) \boldsymbol{M} \left(\widehat{\boldsymbol{M}}^\dagger - \boldsymbol{M}^\dagger \right) (\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}} \\
&= (\boldsymbol{c}^{\mathrm{B}^{>0}})^\top \boldsymbol{B}^{>0} \boldsymbol{M}^{\dagger 1/2} \left(\boldsymbol{M}^{1/2} \widehat{\boldsymbol{M}}^\dagger \boldsymbol{M}^{1/2} - \boldsymbol{I} \right)^2 \boldsymbol{M}^{\dagger 1/2} (\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}} \\
&\leq \left\| \boldsymbol{M}^{1/2} \widehat{\boldsymbol{M}}^\dagger \boldsymbol{M}^{1/2} - \boldsymbol{I} \right\|_2^2 \left\| \boldsymbol{M}^{\dagger 1/2} (\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}} \right\|_2^2.
\end{aligned}$$

The last inequality is by the Courant-Fischer theorem.

By Equation (3.29) and the fact that $\boldsymbol{M}, \widehat{\boldsymbol{M}}$ have same null space,

$$\left(\beta^{-1} - 1\right) \boldsymbol{I} \preccurlyeq \boldsymbol{M}^{1/2} \widehat{\boldsymbol{M}}^\dagger \boldsymbol{M}^{1/2} - \boldsymbol{I} \preccurlyeq \left(\alpha^{-1} - 1\right) \boldsymbol{I}.$$

By Equation (3.30) and (3.31),

$$\left\| \boldsymbol{M}^{1/2} \widehat{\boldsymbol{M}}^\dagger \boldsymbol{M}^{1/2} - \boldsymbol{I} \right\|_2 \leq 2^{-k+2}.$$

Therefore,

$$\|\boldsymbol{y}^* - \boldsymbol{x}^*\|_{\boldsymbol{M}} \leq 2^{-k+2} \left\|(\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_{\boldsymbol{M}^\dagger}.$$

□

We then show the approximate solutions of the two linear systems in Equation (3.27) and (3.28) are close.

Lemma 3.5.3. *Let $\boldsymbol{x}$ be a vector such that*

$$\left\|\boldsymbol{B}^{int}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{B}^{int}}\boldsymbol{c}^{\boldsymbol{B}^{>0}}\right\|_2 \leq \epsilon_2 \left\|\boldsymbol{\Pi}_{\boldsymbol{B}^{int}}\boldsymbol{c}^{\boldsymbol{B}^{>0}}\right\|_2.$$

Then,

$$\left\|\boldsymbol{B}^{>0}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\boldsymbol{B}^{>0}}\right\|_2 \leq \left(\epsilon_2 + 2^{-k+2}(1+\epsilon_2)\right) \left\|\boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\boldsymbol{B}^{>0}}\right\|_2.$$

Proof. Let

$$\boldsymbol{x}^* \stackrel{\text{def}}{=} \boldsymbol{M}^\dagger(\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}} \text{ and } \boldsymbol{y}^* \stackrel{\text{def}}{=} \widehat{\boldsymbol{M}}^\dagger(\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}}.$$

Expanding the left hand side,

$$\begin{aligned}\left\|\boldsymbol{B}^{>0}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2 &= \|\boldsymbol{x} - \boldsymbol{x}^*\|_{\boldsymbol{M}} \\ &\leq \|\boldsymbol{x} - \boldsymbol{y}^*\|_{\boldsymbol{M}} + \|\boldsymbol{y}^* - \boldsymbol{x}^*\|_{\boldsymbol{M}}.\end{aligned}$$

The last inequality is due to the triangle inequality. By Equation (3.29), the first term can be upper bounded by

$$\|\boldsymbol{x} - \boldsymbol{y}^*\|_{\boldsymbol{M}} \leq \alpha^{-1/2} \|\boldsymbol{x} - \boldsymbol{y}^*\|_{\widehat{\boldsymbol{M}}}.$$

The second term is upper bounded by Lemma 3.5.2. Thus, we have

$$\left\|\boldsymbol{B}^{>0}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2 \leq \alpha^{-1/2}\epsilon_2 \left\|\boldsymbol{\Pi}_{\boldsymbol{B}^{int}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2 + 2^{-k+2} \left\|\boldsymbol{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2.$$

Note that

$$\left\|\mathbf{\Pi}_{\boldsymbol{B}^{int}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2 = \left\|\boldsymbol{B}^{>0}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_{\widehat{\boldsymbol{M}}^\dagger} \text{ and } \left\|\mathbf{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2 = \left\|\boldsymbol{B}^{>0}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_{\boldsymbol{M}^\dagger}.$$

Again, by Equation (3.29),

$$\left\|\boldsymbol{B}^{>0}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_{\widehat{\boldsymbol{M}}^\dagger} \leq \alpha^{-1/2}\left\|\boldsymbol{B}^{>0}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_{\boldsymbol{M}^\dagger}.$$

By Equation (3.30) and (3.31),

$$\left\|\boldsymbol{B}^{>0}\boldsymbol{x} - \mathbf{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2 \leq \left(\epsilon_2 + 2^{-k+2}(1+\epsilon_2)\right)\left\|\mathbf{\Pi}_{\boldsymbol{B}^{>0}}\boldsymbol{c}^{\mathrm{B}^{>0}}\right\|_2.$$

This completes the proof. □

Proof of Lemma 3.2.7. We set

$$k \stackrel{\text{def}}{=} \left\lceil \log_2 \frac{s}{\epsilon} \right\rceil + 2.$$

After rounding and scaling, the number of nonzero entries does not change. The value of the smallest nonzero entry does not decrease. The value of the largest entry is multiplied by $2^k = s\epsilon^{-1}$, which is upper bounded by $s\epsilon^{-1}U$.

By Lemma 3.5.1,

$$\kappa(\boldsymbol{B}^{>0,\mathbb{Z}}) \leq (1 + s^{-1/2}\epsilon^{1/2})\kappa(\boldsymbol{B}^{>0}).$$

By Lemma 3.5.3, the accuracy is bounded by $\epsilon/3$. □

3.6 $\mathcal{G}$ Efficiently Reducible to $\mathcal{G}_{\mathbf{z},2}$

In this section, we prove Lemma 3.2.4.

Definition 3.6.1. *We let $\mathcal{G}_z$ denote the class of all matrices with integer valued entries s.t. there is at least one non-zero entry in every row and column, and every row has zero row sum.*

3.6.1 $\mathcal{G} \leq_f \mathcal{G}_z$

In this section, we show the reduction from $\mathcal{G}$ to $\mathcal{G}_z$. Given an instance of linear system $(\boldsymbol{A}, \boldsymbol{c}^A, \epsilon_1)$ with $\boldsymbol{A} \in \mathcal{G}$, the goal is to construct an instance of linear system $(\boldsymbol{A}^Z, \boldsymbol{c}^Z, \epsilon_2)$ with $\boldsymbol{A}^Z \in \mathcal{G}_z$ such that, there exists a map between the solutions of the two linear systems.

Algorithm 8 shows the construction of $(\boldsymbol{A}^Z, \boldsymbol{c}^Z, \epsilon_2)$, and Algorithm 9 shows the transform from a solution of $(\boldsymbol{A}^Z, \boldsymbol{c}^Z, \epsilon_2)$ to a solution of $(\boldsymbol{A}, \boldsymbol{c}^A, \epsilon_1)$. Recall that $\boldsymbol{A}$ has n_A columns. According to the algorithm, $\boldsymbol{A}^Z$ has $(n_A + 1)$ columns, and $\boldsymbol{c}^Z = \boldsymbol{c}$.

Algorithm 8 REDUCE $\mathcal{G}$ TO $\mathcal{G}_z$

Input: $(\boldsymbol{A}, \boldsymbol{c}^A, \epsilon_1)$ where $\boldsymbol{A}_1 \in \mathcal{G}$ is an $m \times n$ matrix, $\boldsymbol{c}^A \in \mathbb{R}^m$, and $0 < \epsilon_1 < 1$.
Output: $(\boldsymbol{A}^Z, \boldsymbol{c}^Z, \epsilon_2)$ where $\boldsymbol{A}^Z \in \mathcal{G}_z$ is an $m' \times n'$ matrix, $\boldsymbol{c}^Z \in \mathbb{R}^{m'}$, and $0 < \epsilon_2 < 1$.
1: **return** $\left(\begin{pmatrix} \boldsymbol{A} & -\boldsymbol{A}\mathbf{1} \end{pmatrix}, \boldsymbol{c}^A, O\left(\frac{\epsilon_1}{n}\right)\right)$.

Algorithm 9 MAPSOLN $\mathcal{G}_z$ TO $\mathcal{G}$

Input: $m \times n$ matrix $\boldsymbol{A} \in \mathcal{G}$, $m' \times n'$ matrix $\boldsymbol{A}^Z \in \mathcal{G}_z$, vector $\boldsymbol{x}^Z \in \mathbb{R}^{n'}$.
Output: Vector $\boldsymbol{x} \in \mathbb{R}^n$.
1: **return** $\boldsymbol{x} \leftarrow \boldsymbol{x}^Z_{1:n} - \boldsymbol{x}^Z_{n+1}\mathbf{1}$

Lemma 3.6.2. *Let $\boldsymbol{A}^Z\boldsymbol{x}^Z = \boldsymbol{c}^Z$ be the linear system returned by a call to* REDUCE $\mathcal{G}$ TO $\mathcal{G}_z(\boldsymbol{A}, \boldsymbol{c}^A, \epsilon_1)$. *Then,*

$$\operatorname{nnz}(\boldsymbol{A}^Z) = O\left(\operatorname{nnz}(\boldsymbol{A})\right),$$

and the largest entry of $\boldsymbol{A}^Z$ is at most $\|\boldsymbol{A}\|_\infty$.

We show the relation between the exact solvers of the two linear systems.

Claim 3.6.3. *Let $\boldsymbol{x}^{Z*} \in \arg\min_{\boldsymbol{x}} \left\|\boldsymbol{A}^Z\boldsymbol{x} - \boldsymbol{c}^Z\right\|_2$. Let $\boldsymbol{x}^*$ be the output vector of Algorithm 9, that is,*

$$\boldsymbol{x}^* = \boldsymbol{x}^{Z*}_{1:n_A} - \boldsymbol{x}^{Z*}_{n_A+1}\mathbf{1}.$$

Then,

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{c}^A \right\|_2 .$$

Proof. Note $\mathbf{1} \in Null(\boldsymbol{A}^{\mathrm{Z}})$, thus,

$$\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^{\mathrm{Z}*} = \boldsymbol{A}^{\mathrm{Z}} \left(\boldsymbol{x}^{\mathrm{Z}*} - \boldsymbol{x}^{\mathrm{Z}*}_{n_A+1}\mathbf{1} \right).$$

Expanding it gives

$$\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^{\mathrm{Z}*} = \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{A}\mathbf{1} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}^* \\ \mathbf{0} \end{pmatrix} = \boldsymbol{A}\boldsymbol{x}^*.$$

Note $\boldsymbol{c}^{\mathrm{Z}} = \boldsymbol{c}$. Therefore, $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{A}} \right\|_2$. □

We then show the relation between the approximate solvers.

Lemma 3.6.4. *Let $\boldsymbol{x}^{\mathrm{Z}}$ be a vector such that $\left\| \boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^{\mathrm{Z}} - \boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^A \right\|_2 \leq \epsilon_2 \left\| \boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^A \right\|_2$. Let $\boldsymbol{x}$ be the output vector of Algorithm 9, that is, $\boldsymbol{x} = \boldsymbol{x}^{\mathrm{Z}}_{1:n_A} - \boldsymbol{x}^{\mathrm{Z}}_{n_A+1}\mathbf{1}$. Then,*

$$\left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^A \right\|_2 \leq O\left(\frac{\sqrt{n_A}\sigma_{\min}(\boldsymbol{A})}{\sigma_{\min}(\boldsymbol{A}^{\mathrm{Z}})} \right) \epsilon_2 \left\| \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^A \right\|_2 .$$

Proof. Let $\boldsymbol{x}^{\mathrm{Z}*} \in \arg\min_{\boldsymbol{x}} \left\| \boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}} \right\|$ whose last entry is 0. Expanding the left hand side norm,

$$\left\| \boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^{\mathrm{Z}} - \boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{A}} \right\|_2 = \left\| \boldsymbol{A}^{\mathrm{Z}} \left(\boldsymbol{x}^{\mathrm{Z}} - \boldsymbol{x}^{\mathrm{Z}*} \right) \right\|_2 .$$

By Claim 3.6.3, $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{A}} \right\|_2$. Thus,

$$\left\| \boldsymbol{A} \left(\boldsymbol{x} - \boldsymbol{x}^* \right) \right\|_2 = \left\| \boldsymbol{A}^{\mathrm{Z}} (\boldsymbol{x}^{\mathrm{Z}} - \boldsymbol{x}^{\mathrm{Z}*}) \right\|_2 .$$

Note that

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^{\mathrm{Z}} \text{ and } \boldsymbol{A}\boldsymbol{x}^{*} = \boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^{\mathrm{Z}*}.$$

Together with the Lemma condition, this gives

$$\left\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2 = \left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^{\mathrm{Z}} - \boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{Z}}\right\|_2 \leq \epsilon_2 \left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{A}}\right\|_2. \tag{3.32}$$

It remains to upper bound $\left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{A}}\right\|_2$ by a function of $\left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2$. Note

$$\left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2 = \left\|\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_{\left(\boldsymbol{A}^{\top}\boldsymbol{A}\right)^{\dagger}} \text{ and } \left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{A}}\right\|_2 = \left\|(\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_{\left((\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{A}^{\mathrm{Z}}\right)^{\dagger}}. \tag{3.33}$$

It suffices to work on $\left\|\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_{\left(\boldsymbol{A}^{\top}\boldsymbol{A}\right)^{\dagger}}$ and $\left\|(\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_{\left((\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{A}^{\mathrm{Z}}\right)^{\dagger}}$. Since $\boldsymbol{A}^{\mathrm{Z}} = \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{A}\boldsymbol{1} \end{pmatrix}$, we have

$$\left\|(\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 = \left\|\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 + \left\|\boldsymbol{1}^{\top}\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2.$$

By Cauchy-Schwarz inequality,

$$\left\|\boldsymbol{1}^{\top}\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2 \leq \|\boldsymbol{1}\|_2 \left\|\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2 = \sqrt{n_A}\left\|\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2.$$

Plugging this gives

$$\left\|(\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 \leq (n_A + 1)\left\|\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2.$$

By Courant-Fischer theorem,

$$\left\|(\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|^2_{\left((\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{A}^{\mathrm{Z}}\right)^{\dagger}} \leq \lambda_{\max}\left(\left((\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{A}^{\mathrm{Z}}\right)^{\dagger}\right)\left\|(\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 \leq \lambda_{\min}^{-1}\left((\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{A}^{\mathrm{Z}}\right)\left\|(\boldsymbol{A}^{\mathrm{Z}})^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2$$

and since $\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}$ is in the eigenspace of $\boldsymbol{A}^{\top}\boldsymbol{A}$,

$$\left\|\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|^2_{\left(\boldsymbol{A}^{\top}\boldsymbol{A}\right)^{\dagger}} \geq \lambda_{\min}\left(\left(\boldsymbol{A}^{\top}\boldsymbol{A}\right)^{\dagger}\right)\left\|\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 = \lambda_{\max}^{-1}\left(\boldsymbol{A}^{\top}\boldsymbol{A}\right)\left\|\boldsymbol{A}^{\top}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2.$$

Combining the above two inequalities,

$$\left\|(\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{c}^{\mathrm{A}}\right\|^2_{((\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}})^\dagger} \leq \frac{\lambda_{\max}\left(\boldsymbol{A}^\top \boldsymbol{A}\right)}{\lambda_{\min}\left((\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}}\right)}(n_A+1)\left\|\boldsymbol{A}^\top \boldsymbol{c}^{\mathrm{A}}\right\|^2_{(\boldsymbol{A}^\top \boldsymbol{A})^\dagger}.$$

Given Equation (3.32) and (3.33), and condition $\left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^{\mathrm{Z}} - \boldsymbol{c}^{\mathrm{A}}\right\|_2 \leq \epsilon \left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{A}}\right\|_2$, we have

$$\left\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2 \leq \epsilon_2 \frac{\sigma_{\max}(\boldsymbol{A})}{\sigma_{\min}(\boldsymbol{A}^{\mathrm{Z}})}\sqrt{n_A+1}\left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2.$$

This completes the proof. □

We compute the nonzero condition number of $\boldsymbol{A}^{\mathrm{Z}}$. Note

$$(\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}} = \begin{pmatrix} \boldsymbol{M} & -\boldsymbol{M}\mathbf{1} \\ -\mathbf{1}^\top \boldsymbol{M} & \mathbf{1}^\top \boldsymbol{M}\mathbf{1} \end{pmatrix}, \tag{3.34}$$

where $\boldsymbol{M} = \boldsymbol{A}^\top \boldsymbol{A}$.

Claim 3.6.5. $\lambda_{\max}\left((\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}}\right) \leq 2n_A \lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A})$.

Proof. Let λ_1 be the largest eigenvalue of $(\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}}$, and $\boldsymbol{y} = (\tilde{\boldsymbol{y}}; \alpha)$ be the associated eigenvector of unit length. Let μ_1 be the largest eigenvalue of $\boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{M}$.

$$\begin{aligned}\lambda_1 = \boldsymbol{y}^\top (\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}} \boldsymbol{y} &= \begin{pmatrix} \tilde{\boldsymbol{y}}^\top & \alpha \end{pmatrix} \begin{pmatrix} \boldsymbol{M} & -\boldsymbol{M}\mathbf{1} \\ -\mathbf{1}^\top \boldsymbol{M} & \mathbf{1}^\top \boldsymbol{M}\mathbf{1} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{y}} \\ \alpha \end{pmatrix} \\ &= \tilde{\boldsymbol{y}}^\top \boldsymbol{M}\tilde{\boldsymbol{y}} - 2\alpha \mathbf{1}^\top \boldsymbol{M}\tilde{\boldsymbol{y}} + \alpha^2 \mathbf{1}^\top \boldsymbol{M}\mathbf{1}.\end{aligned}$$

By Courant-Fischer Theorem,

$$\tilde{\boldsymbol{y}}^\top \boldsymbol{M}\tilde{\boldsymbol{y}} \leq \mu_1 \tilde{\boldsymbol{y}}^\top \tilde{\boldsymbol{y}}, \text{ and } \mathbf{1}^\top \boldsymbol{M}\mathbf{1} \leq \mu_1 n_A.$$

By Cauchy-Schwarz inequality,

$$\left|\mathbf{1}^\top \boldsymbol{M}\tilde{\boldsymbol{y}}\right| \leq \|\mathbf{1}\|_2 \|\boldsymbol{M}\tilde{\boldsymbol{y}}\|_2 \leq \sqrt{n_A}\mu_1 \|\tilde{\boldsymbol{y}}\|_2 .$$

Putting all together,

$$\begin{aligned}\lambda_1 &\leq \mu_1 \left(\|\tilde{\boldsymbol{y}}\|_2^2 + 2|\alpha|\sqrt{n_A}\|\tilde{\boldsymbol{y}}\|_2 + \alpha^2 n_A\right) \\ &= \mu_1 \left(\|\tilde{\boldsymbol{y}}\|_2 + |\alpha|\sqrt{n_A}\right)^2 .\end{aligned}$$

Since $\|\tilde{\boldsymbol{y}}\|_2 , |\alpha| \leq 1$, we have

$$\lambda_1 \leq 2\mu_1 n_A.$$

This completes the proof. □

Before lower bounding the smallest nonzero singular value of $\boldsymbol{A}^{\mathrm{Z}}$, we characterize the null space of $\boldsymbol{A}^{\mathrm{Z}}$ by the following Claim.

Claim 3.6.6.

$$\mathrm{null}(\boldsymbol{A}^{\mathrm{Z}}) = \textbf{\textit{Span}}\left(\mathbf{1}, \begin{pmatrix} \boldsymbol{z} \\ 0 \end{pmatrix} : \boldsymbol{z} \in \mathrm{null}(\boldsymbol{A})\right).$$

Proof. Let $\boldsymbol{S}$ be the subspace of $\textbf{Span}\left(\mathbf{1}, \begin{pmatrix} \boldsymbol{z} \\ 0 \end{pmatrix} : \boldsymbol{z} \in \mathrm{null}(\boldsymbol{A})\right)$.

We first show that $\boldsymbol{S} \subseteq \mathrm{null}(\boldsymbol{A}^{\mathrm{Z}})$. Clearly,

$$\boldsymbol{A}^{\mathrm{Z}}\mathbf{1} = \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{A}\mathbf{1} \end{pmatrix}\mathbf{1} = \mathbf{0}.$$

For each $\boldsymbol{z} \in \mathrm{null}(\boldsymbol{A})$, we have

$$\boldsymbol{A}^{\mathrm{Z}}\begin{pmatrix} \boldsymbol{z} \\ 0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{A}\mathbf{1} \end{pmatrix}\begin{pmatrix} \boldsymbol{z} \\ 0 \end{pmatrix} = \boldsymbol{A}\boldsymbol{z} = \mathbf{0}.$$

Thus, $\boldsymbol{S} \subseteq \mathrm{null}(\boldsymbol{A}^{\mathrm{Z}})$.

We then show that $\boldsymbol{S} \supseteq \mathrm{null}(\boldsymbol{A}^{\mathrm{Z}})$. For any $\boldsymbol{z}' \in \mathrm{null}(\boldsymbol{A}^{\mathrm{Z}})$, $\boldsymbol{z}' - \boldsymbol{z}'_{n_A+1}\mathbf{1} \in \mathrm{null}(\boldsymbol{A}^{\mathrm{Z}})$. Let $\boldsymbol{z} \in \mathbb{R}^n$ such that the ith entry of $\boldsymbol{z}$ is $\boldsymbol{z}'_i - \boldsymbol{z}'_{n_A+1}$. Thus,

$$\mathbf{0} = \boldsymbol{A}^{\mathrm{Z}}\left(\boldsymbol{z}' - \boldsymbol{z}'_{n_A+1}\mathbf{1}\right) = \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{A}\mathbf{1} \end{pmatrix} \begin{pmatrix} \boldsymbol{z} \\ 0 \end{pmatrix} = \boldsymbol{A}\boldsymbol{z}.$$

That is, any vector in $\boldsymbol{S}$ can be written as a linear combination of $\mathbf{1}$ and $\begin{pmatrix} \boldsymbol{z} \\ 0 \end{pmatrix}$ where $\boldsymbol{z} \in \mathrm{null}(\boldsymbol{A})$. Thus, $\boldsymbol{S} \supseteq \mathrm{null}(\boldsymbol{A}^{\mathrm{Z}})$.

Therefore, $\mathrm{null}(\boldsymbol{A}^{\mathrm{Z}}) = \boldsymbol{S}$. □

By the above claim, we know that $\boldsymbol{A}$ and $\boldsymbol{A}^{\mathrm{Z}}$ have same rank. We bound the smallest nonzero singular value of $\boldsymbol{A}^{\mathrm{Z}}$ by the following Claim.

Claim 3.6.7. $\lambda_{\min}\left((\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}}\right) \geq \lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A})/(n_A + 1)$.

Proof. Let λ_k be the smallest nonzero eigenvalue of $(\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}}$, and $\boldsymbol{y} = (\tilde{\boldsymbol{y}}; \alpha)$ be the associated eigenvector of unit length. Let μ_k be the smallest nonzero eigenvalue of $\boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{M}$, and $\boldsymbol{x}$ be the associated eigenvector of unit length.

$$\lambda_k = \tilde{\boldsymbol{y}}^\top \boldsymbol{M}\tilde{\boldsymbol{y}} - 2\alpha\mathbf{1}^\top \boldsymbol{M}\tilde{\boldsymbol{y}} + \alpha^2 \mathbf{1}^\top \boldsymbol{M}\mathbf{1}.$$

Since $\boldsymbol{M}$ is symmetric PSD, it can be written as $\boldsymbol{M} = \boldsymbol{M}^{1/2}\boldsymbol{M}^{1/2}$.

$$\lambda_k = \left\| \boldsymbol{M}^{1/2}\left(\tilde{\boldsymbol{y}} - \alpha\mathbf{1}\right) \right\|_2^2.$$

We decompose $\tilde{\boldsymbol{y}} - \alpha\mathbf{1} := \boldsymbol{z}_1 + \boldsymbol{z}_2$, where $\boldsymbol{z}_1 \in \mathrm{null}(\boldsymbol{M})$ and $\boldsymbol{z}_2 \perp \mathrm{null}(\boldsymbol{M})$. Then, $\lambda_k = \left\| \boldsymbol{M}^{1/2}\boldsymbol{z}_2 \right\|_2^2$. By Courant-Fischer Theorem

$$\lambda_k \geq \mu_k \left\| \boldsymbol{z}_2 \right\|_2^2.$$

To lower bound λ_k, it suffices to lower bound $\|\boldsymbol{z}_2\|_2$.

Since $\boldsymbol{y} = (\tilde{\boldsymbol{y}}; \alpha)$,

$$\|\tilde{\boldsymbol{y}} - \alpha\mathbf{1}\|_2^2 = \left\| \boldsymbol{y} - \alpha \begin{pmatrix} \mathbf{1} \\ 1 \end{pmatrix} \right\|_2^2 .$$

Since $\boldsymbol{y} \perp \mathbf{1}$, which is in $\mathrm{null}(\boldsymbol{A}^{\mathrm{Z}})$, we have

$$\|\tilde{\boldsymbol{y}} - \alpha\mathbf{1}\|_2^2 = \|\boldsymbol{y}\|_2^2 + \alpha^2(1 + n_A) = \|\tilde{\boldsymbol{y}}\|_2^2 + \alpha^2(2 + n_A).$$

Vectors $\tilde{\boldsymbol{y}}, -\alpha\mathbf{1}, \tilde{\boldsymbol{y}} - \alpha\mathbf{1}$ form a triangle. Let θ be the angle between $\tilde{\boldsymbol{y}}$ and $\alpha\mathbf{1}$. See Figure 3.1.

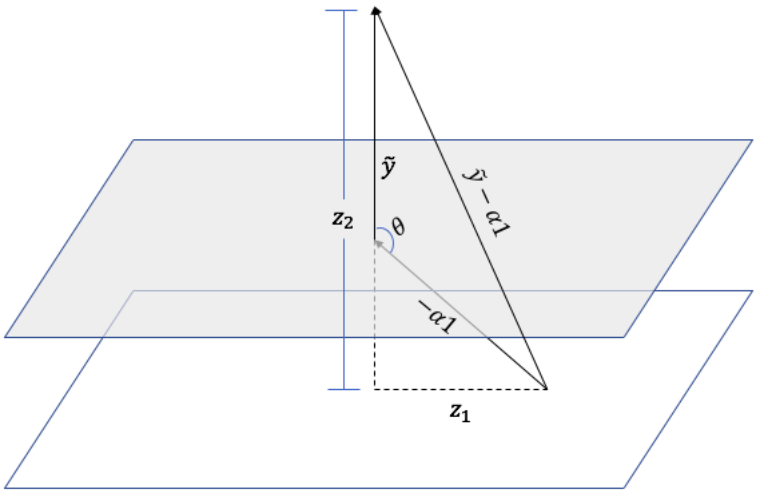

Figure 3.1: The relation between vectors $\tilde{\boldsymbol{y}}, -\alpha\mathbf{1}$ and $\tilde{\boldsymbol{y}} - \alpha\mathbf{1}$.

Then,

$$\cos(\theta) = \frac{\|\tilde{\boldsymbol{y}}\|_2^2 + \|\alpha\mathbf{1}\|_2^2 - \|\tilde{\boldsymbol{y}} - \alpha\mathbf{1}\|_2^2}{2\,\|\tilde{\boldsymbol{y}}\|_2\,\|\alpha\mathbf{1}\|_2} = -\frac{|\alpha|}{\|\tilde{\boldsymbol{y}}\|_2\sqrt{n_A}}.$$

Thus,

$$\|\boldsymbol{z}_2\|_2 = \|\tilde{\boldsymbol{y}}\|_2 + \|\alpha\mathbf{1}\|_2 \cos(\pi - \theta) = \|\tilde{\boldsymbol{y}}\|_2 + \frac{\alpha^2}{\|\tilde{\boldsymbol{y}}\|_2} \geq \|\tilde{\boldsymbol{y}}\|_2 .$$

On the other hand,

$$1 = \alpha^2 + \|\tilde{\boldsymbol{y}}\|_2^2 = \left(\sum_i \tilde{\boldsymbol{y}}_i\right)^2 + \sum_i \tilde{\boldsymbol{y}}_i^2 \leq (n_A + 1) \sum_i \tilde{\boldsymbol{y}}_i^2 = (n_A + 1) \|\tilde{\boldsymbol{y}}\|_2^2 .$$

Therefore,

$$\lambda_k \geq \mu_k \|\tilde{\boldsymbol{y}}\|_2^2 \geq \frac{\mu_k}{n_A + 1}.$$

This completes the proof. □

The above lemmas give an upper bound on the condition number of $\boldsymbol{A}^{\mathrm{Z}}$.

Lemma 3.6.8. $\kappa(\boldsymbol{A}^{\mathrm{Z}}) \leq O(n_A^{3/2})\kappa(\boldsymbol{A})$.

3.6.2 $\mathcal{G}_{\mathrm{z}} \leq_f \mathcal{G}_{\mathrm{z},2}$

In this section, we show the reduction from $\mathcal{G}_{\mathrm{z}}$ to $\mathcal{G}_{\mathrm{z},2}$. Given an instance of linear system $(\boldsymbol{A}^{\mathrm{Z}}, \boldsymbol{c}^{\mathrm{Z}}, \epsilon_1)$ with $\boldsymbol{A}^{\mathrm{Z}} \in \mathcal{G}_{\mathrm{z}}$, the goal is to construct an instance of linear system $(\boldsymbol{A}^{\mathrm{Z},2}, \boldsymbol{c}^{\mathrm{Z},2}, \epsilon_2)$ with $\boldsymbol{A}^{\mathrm{Z},2} \in \mathcal{G}_{\mathrm{z},2}$ such that, there is a map between the solutions of these two linear systems.

Algorithm 10 shows the construction of $(\boldsymbol{A}^{\mathrm{Z},2}, \boldsymbol{c}^{\mathrm{Z},2}, \epsilon_2)$, and Algorithm 11 shows the transform from a solution of $(\boldsymbol{A}^{\mathrm{Z},2}, \boldsymbol{c}^{\mathrm{Z},2}, \epsilon_2)$ to a solution of $(\boldsymbol{A}^{\mathrm{Z}}, \boldsymbol{c}^{\mathrm{Z}}, \epsilon_1)$. Note $\boldsymbol{c}^{\mathrm{Z},2} = (\boldsymbol{c}^{\mathrm{Z}}; 0)$.

Lemma 3.6.9. *Let $\boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x}^{\mathrm{Z},2} = \boldsymbol{c}^{\mathrm{Z},2}$ be the linear system returned by a call to* REDUCE $\mathcal{G}_z$TO$\mathcal{G}_{z,2}(\boldsymbol{A}^{\mathrm{Z}}, \boldsymbol{c}^{\mathrm{Z}}, \epsilon_1)$. *Then,*

$$\mathrm{nnz}(\boldsymbol{A}^{\mathrm{Z},2}) = O\left(\mathrm{nnz}(\boldsymbol{A}^{\mathrm{Z}})\right).$$

and the largest entry of $\boldsymbol{A}^{\mathrm{Z},2}$ is at most $\epsilon_1^{-1}\sqrt{m}\sigma_{\max}(\boldsymbol{A}^{\mathrm{Z}}) \left\|\boldsymbol{A}^{\mathrm{Z}}\right\|_\infty \|\boldsymbol{c}^{\mathrm{Z}}\|_2$.

Algorithm 10 REDUCE $\mathcal{G}_{\text{z}}$ TO $\mathcal{G}_{\text{z,2}}$

Input: $(\boldsymbol{A}^{\text{Z}}, \boldsymbol{c}^{\text{Z}}, \epsilon_1)$ where $\boldsymbol{A}^{\text{Z}} \in \mathcal{G}_{\text{z}}$ is an $m \times n$ matrix, $\boldsymbol{c}^{\text{Z}} \in \mathbb{R}^m$, and $0 < \epsilon_1 < 1$.

Output: $(\boldsymbol{A}^{\text{Z,2}}, \boldsymbol{c}^{\text{Z,2}}, \epsilon_2)$ where $\boldsymbol{A}^{\text{Z,2}} \in \mathcal{G}_{\text{z,2}}$ is an $m' \times n'$ matrix, $\boldsymbol{c}^{\text{Z,2}} \in \mathbb{R}^{m'}$, and $0 < \epsilon_2 < 1$.

$w \leftarrow 3\epsilon_1^{-1}\sqrt{m}\sigma_{\max}(\boldsymbol{A}^{\text{Z}}) \left\|\boldsymbol{A}^{\text{Z}}\right\|_\infty \left\|\boldsymbol{c}^{\text{Z}}\right\|_2$

$k^* \leftarrow \min\left\{k \in \mathbb{Z} : 2^k \geq \left\|\boldsymbol{A}^{\text{Z}}\right\|_\infty\right\}$

Let $\boldsymbol{a} \in \mathbb{R}^n$

for $i \leftarrow 1$ to n **do**

  $\boldsymbol{a}_i \leftarrow 2^{k^*} - \left\|\boldsymbol{A}_i^{\text{Z}}\right\|_1 / 2$

end for

$\boldsymbol{A}^{\text{Z,2}} = \begin{pmatrix} \boldsymbol{A}^{\text{Z}} & \boldsymbol{a} & -\boldsymbol{a} \\ \boldsymbol{0} & w & -w \end{pmatrix}$

$\boldsymbol{c}^{\text{Z,2}} \leftarrow \begin{pmatrix} \boldsymbol{c}^{\text{Z}} \\ 0 \end{pmatrix}$

return $\left(\boldsymbol{A}^{\text{Z,2}}, \boldsymbol{c}^{\text{Z,2}}, O\left(\frac{\epsilon_1}{\sigma_{\max}(\boldsymbol{A}^{\text{Z}})\|\boldsymbol{c}^{\text{Z}}\|_2}\right)\right)$.

Algorithm 11 MAPSOLN $\mathcal{G}_{\text{z,2}}$ TO $\mathcal{G}_{\text{z}}$

Input: $m \times n$ matrix $\boldsymbol{A}^{\text{Z}} \in \mathcal{G}_{\text{z}}$, $m' \times n'$ matrix $\boldsymbol{A}^{\text{Z,2}} \in \mathcal{G}_{\text{z,2}}$, vector $\boldsymbol{c}^{\text{Z}} \in \mathbb{R}^m$, vector $\boldsymbol{x} \in \mathbb{R}^{n'}$.

Output: Vector $\boldsymbol{y} \in \mathbb{R}^n$.

if $(\boldsymbol{A}^{\text{Z}})^\top \boldsymbol{c}^{\text{Z}} = \boldsymbol{0}$ **then**

  return $\boldsymbol{y} \leftarrow \boldsymbol{0}$

else

  return $\boldsymbol{y} \leftarrow \boldsymbol{x}_{1:n}$

end if

Claim 3.6.10. $\text{null}(\boldsymbol{A}^{\text{Z,2}}) = \textbf{\textit{Span}}\left(\left\{\begin{pmatrix} \boldsymbol{0} \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} \boldsymbol{x} \\ 0 \\ 0 \end{pmatrix} : \boldsymbol{x} \in \text{null}(\boldsymbol{A}^{\text{Z}})\right\}\right)$.

Proof. Let $\boldsymbol{y} = (\boldsymbol{x}; \alpha; \beta) \in \mathbb{R}^{n_A+2}$ such that $\boldsymbol{y} \in \text{null}(\boldsymbol{A}^{\text{Z,2}})$, that is,

$$\boldsymbol{A}^{\text{Z,2}}\boldsymbol{y} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{a} & -\boldsymbol{a} \\ \boldsymbol{0} & w & -w \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}^{\text{Z}}\boldsymbol{x} + (\alpha - \beta)\boldsymbol{a} \\ (\alpha - \beta)w \end{pmatrix} = \boldsymbol{0}.$$

Since $w > 0$, we have

$$\alpha = \beta.$$

and thus

$$\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} = \mathbf{0}.$$

Therefore,

$$\mathrm{null}(\boldsymbol{A}^{\mathrm{Z},2}) = \textbf{Span}\left(\left\{\begin{pmatrix}\mathbf{0}\\1\\1\end{pmatrix}, \begin{pmatrix}\boldsymbol{x}\\0\\0\end{pmatrix} : \boldsymbol{x} \in \mathrm{null}(\boldsymbol{A}^{\mathrm{Z}})\right\}\right).$$

This completes the proof. □

We write $\boldsymbol{x}^{\mathrm{Z},2}$ as $(\boldsymbol{x}; \alpha, \beta)$. By Claim 3.6.10, the vector $(\mathbf{0}; 1; 1)$ is in the null space of $\boldsymbol{A}^{\mathrm{Z},2}$. Thus,

$$\boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x}^{\mathrm{Z},2} = \boldsymbol{A}^{\mathrm{Z},2}\left(\boldsymbol{x}^{\mathrm{Z},2} - \frac{\alpha+\beta}{2}\begin{pmatrix}\mathbf{0}\\1\\1\end{pmatrix}\right) = \boldsymbol{A}^{\mathrm{Z},2}\begin{pmatrix}\boldsymbol{x}\\\frac{\alpha-\beta}{2}\\\frac{\beta-\alpha}{2}\end{pmatrix}.$$

Without loss of generality, we assume $\alpha = \beta$, that is, $\boldsymbol{x}^{\mathrm{Z},2} = (\boldsymbol{x}; \alpha; -\alpha)$.

We first show that the exact solutions of the two linear systems are close.

Lemma 3.6.11. *Let* $\boldsymbol{x}^{\mathrm{Z},2*} \in \arg\min_{\boldsymbol{x}} \left\|\boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z},2}\right\|_2$. *Write* $\boldsymbol{x}^{\mathrm{Z},2*} = (\boldsymbol{s}^*; \alpha; -\alpha)$. *Then,*

$$\boldsymbol{s}^* \in \arg\min_{\boldsymbol{x}} \left\|\boldsymbol{A}^{Z}\boldsymbol{x} - \boldsymbol{c}^{Z}\right\|_{\boldsymbol{P}},$$

where $\boldsymbol{P} = \boldsymbol{I} - \frac{\boldsymbol{a}\boldsymbol{a}^\top}{w^2+\|\boldsymbol{a}\|_2^2}$.

Proof. Without loss of generality, we write $\boldsymbol{x}^{\mathrm{Z},2} = (\boldsymbol{x}; \alpha; -\alpha)$. We expand $\boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x}^{\mathrm{Z},2} - \boldsymbol{c}^{\mathrm{Z},2}$,

$$
\begin{aligned}
&\min_{\boldsymbol{x}^{\mathrm{Z},2}} \left\| \boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x}^{\mathrm{Z},2} - \boldsymbol{c}^{\mathrm{Z},2} \right\|_2^2 \\
&= \min_{\boldsymbol{x},\alpha} \left\| \begin{pmatrix} \boldsymbol{A}^{\mathrm{Z}} & \boldsymbol{a} & -\boldsymbol{a} \\ \boldsymbol{0} & w & -w \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \alpha \\ -\alpha \end{pmatrix} - \begin{pmatrix} \boldsymbol{c}^{\mathrm{Z}} \\ 0 \end{pmatrix} \right\|_2^2 \\
&= \min_{\boldsymbol{x},\alpha} \left\| \boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}} + 2\alpha\boldsymbol{a} \right\|_2^2 + 4w^2\alpha^2 \\
&= \min_{\boldsymbol{x}} \min_{\alpha} 4\left(w^2 + \|\boldsymbol{a}\|_2^2\right) \left(\alpha + \frac{\boldsymbol{a}^\top(\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}})}{2\left(w^2 + \|\boldsymbol{a}\|_2^2\right)} \right)^2 + \left\| \boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}} \right\|_2^2 - \frac{\left(\boldsymbol{a}^\top(\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}})\right)^2}{w^2 + \|\boldsymbol{a}\|_2^2}.
\end{aligned}
$$

Since the minimization over $\boldsymbol{x}$ and α is independent of each other, for any fixed $\boldsymbol{x}$, the above value is minimized when

$$
\alpha = -\frac{\boldsymbol{a}^\top(\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}})}{2\left(w^2 + \|\boldsymbol{a}\|_2^2\right)}.
$$

Plugging this value of α gives

$$
\begin{aligned}
\min_{\boldsymbol{x}^{\mathrm{Z},2}} \left\| \boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x}^{\mathrm{Z},2} - \boldsymbol{c}^{\mathrm{Z},2} \right\|_2^2 &= \min_{\boldsymbol{x}} \left\| \boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}} \right\|_2^2 - \frac{\left(\boldsymbol{a}^\top(\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}})\right)^2}{w^2 + \|\boldsymbol{a}\|_2^2} \\
&= \min_{\boldsymbol{x}} \left(\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}}\right)^\top \left(\boldsymbol{I} - \frac{\boldsymbol{a}\boldsymbol{a}^\top}{w^2 + \|\boldsymbol{a}\|_2^2} \right) \left(\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}}\right) \\
&= \min_{\boldsymbol{x}} \left\| \boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}} \right\|_{\boldsymbol{P}}^2,
\end{aligned}
$$

where $\boldsymbol{P} = \boldsymbol{I} - \frac{\boldsymbol{a}\boldsymbol{a}^\top}{w^2 + \|\boldsymbol{a}\|_2^2}$. The fact $\boldsymbol{x}^{\mathrm{Z},2*} \in \arg\min_{\boldsymbol{x}} \left\| \boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z},2} \right\|_2$ implies that

$$
\boldsymbol{s}^* \in \arg\min_{\boldsymbol{x}} \left\| \boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}} \right\|_{\boldsymbol{P}},
$$

which completes the proof. □

Lemma 3.6.12. *Let* $\boldsymbol{x}^{\mathrm{Z},2*} \in \arg\min_{\boldsymbol{x}} \left\| \boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z},2} \right\|_2$ *such that* $\boldsymbol{x}^{\mathrm{Z},2*}$ *has the form* $(\boldsymbol{s}^*; \alpha; -\alpha)$

and $\left\|\boldsymbol{x}^{\mathrm{Z},2*}\right\|_2$ *is minimized. Let* $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}} \left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}}\right\|_2$ *such that* $\left\|\boldsymbol{x}^*\right\|_2$ *is minimized. Then,*

$$\left\|\boldsymbol{A}^{\mathrm{Z}}\left(\boldsymbol{s}^* - \boldsymbol{x}^*\right)\right\|_2 \leq \frac{\left\|\boldsymbol{a}\right\|_2}{w}\sigma_{\max}(\boldsymbol{A}^{\mathrm{Z}})\left\|\boldsymbol{c}^{\mathrm{Z}}\right\|_2\left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{Z}}\right\|_2.$$

Proof. Let $\gamma \stackrel{\text{def}}{=} \frac{\left\|\boldsymbol{a}\right\|_2^2}{w^2+\left\|\boldsymbol{a}\right\|_2^2}$. By the definition of $\boldsymbol{P}$ in Lemma 3.6.11,

$$(1-\gamma)\,\boldsymbol{I} \preccurlyeq \boldsymbol{P} \preccurlyeq \boldsymbol{I}.$$

Thus,

$$(1-\gamma)\left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{s}^* - \boldsymbol{c}^{\mathrm{Z}}\right\|_2^2 \leq \left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{s}^* - \boldsymbol{c}^{\mathrm{Z}}\right\|_{\boldsymbol{P}}^2 \leq \left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^* - \boldsymbol{c}^{\mathrm{Z}}\right\|_{\boldsymbol{P}}^2 \leq \left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^* - \boldsymbol{c}^{\mathrm{Z}}\right\|_2^2. \quad (3.35)$$

The second inequality is due to $\boldsymbol{s}^* \in \arg\min_{\boldsymbol{x}} \left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}}\right\|_{\boldsymbol{P}}$. Note that $\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^* - \boldsymbol{c}^{\mathrm{Z}}$ is orthogonal to the column space of $\boldsymbol{A}^{\mathrm{Z}}$, in particular, it is orthogonal to $\boldsymbol{A}^{\mathrm{Z}}(\boldsymbol{s}^* - \boldsymbol{x}^*)$.

$$\left\|\boldsymbol{A}^{\mathrm{Z}}(\boldsymbol{s}^* - \boldsymbol{x}^*)\right\|_2^2 = \left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{s}^* - \boldsymbol{c}^{\mathrm{Z}}\right\|_2^2 - \left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^* - \boldsymbol{c}^{\mathrm{Z}}\right\|_2^2.$$

By Equation (3.35),

$$\left\|\boldsymbol{A}^{\mathrm{Z}}(\boldsymbol{s}^* - \boldsymbol{x}^*)\right\|_2^2 \leq \frac{\gamma}{1-\gamma}\left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x}^* - \boldsymbol{c}^{\mathrm{Z}}\right\|_2^2 \leq \frac{\gamma}{1-\gamma}\left\|\boldsymbol{c}^{\mathrm{Z}}\right\|_2^2.$$

If $\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{Z}} = \boldsymbol{0}$, then $\boldsymbol{s}^* = \boldsymbol{x}^* = \boldsymbol{0}$ and the claim holds. Otherwise, by Lemma 2.2.5,

$$\left\|\boldsymbol{A}^{\mathrm{Z}}(\boldsymbol{s}^* - \boldsymbol{x}^*)\right\|_2 \leq \sqrt{\frac{\gamma}{1-\gamma}}\sigma_{\max}(\boldsymbol{A}^{\mathrm{Z}})\left\|\boldsymbol{c}^{\mathrm{Z}}\right\|_2\left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{Z}}\right\|_2.$$

Plugging the value of γ completes the proof. □

We then show the approximate solvers of the two linear systems are close.

Lemma 3.6.13. *Let* ϵ_2 *be the error parameter returned by a call to* REDUCE $\mathcal{G}_z$TO$\mathcal{G}_{z,2}(\boldsymbol{A}^{\mathrm{Z}}, \boldsymbol{c}^{\mathrm{Z}}, \epsilon_1)$

(Algorithm 10). Let $\boldsymbol{x}^{\mathrm{Z},2}$ *be a vector such that* $\left\|\boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x}^{\mathrm{Z},2} - \boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z},2}}\boldsymbol{c}^{\mathrm{Z},2}\right\|_2 \leq \epsilon_2 \left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z},2}}\boldsymbol{c}^{\mathrm{Z},2}\right\|_2$. *Let* $\boldsymbol{x}$ *be the vector returned by Algorithm 11. Then,*

$$\left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{Z}}\right\|_2 \leq \epsilon_1 \left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{Z}}\right\|_2.$$

Proof. If $\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{Z}} = \boldsymbol{0}$, then $\boldsymbol{x} = \boldsymbol{0}$ and the statement holds. If $\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}}\boldsymbol{c}^{\mathrm{Z}} \neq \boldsymbol{0}$, then $\boldsymbol{x} = \boldsymbol{x}^{\mathrm{Z},2}_{1:n_A+1}$. Let $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}} \left\|\boldsymbol{A}^{\mathrm{Z}}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z}}\right\|_2$. We now bound the difference between our solution $\boldsymbol{x}$ and $\boldsymbol{x}^*$. Let $\boldsymbol{x}^{\mathrm{Z},2*} \in \arg\min_{\boldsymbol{x}} \left\|\boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{Z},2}\right\|_2$ of the form $(\boldsymbol{s}^*; \alpha; -\alpha)$. By the triangle inequality,

$$\left\|\boldsymbol{A}^{\mathrm{Z}}(\boldsymbol{x} - \boldsymbol{x}^*)\right\|_2 \leq \left\|\boldsymbol{A}^{\mathrm{Z}}(\boldsymbol{x} - \boldsymbol{s}^*)\right\|_2 + \left\|\boldsymbol{A}^{\mathrm{Z}}(\boldsymbol{s}^* - \boldsymbol{x}^*)\right\|_2. \tag{3.36}$$

The second term is upper bound by Lemma 3.6.12. It remains to upper bound the first term. Let

$$\delta \stackrel{\text{def}}{=} \epsilon_2 \left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z},2}}\boldsymbol{c}^{\mathrm{Z},2}\right\|_2. \tag{3.37}$$

Without loss of generality, we write $\boldsymbol{x}^{\mathrm{Z},2}$ as $(\boldsymbol{x}; \beta; -\beta)$, where $\boldsymbol{x}$ is the output of Algo-

rithm 11.

$$\begin{aligned}
\left\|\boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x}^{\mathrm{Z},2}-\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z},2}}\boldsymbol{c}^{\mathrm{Z},2}\right\|_2^2 &= \left\|\boldsymbol{A}^{\mathrm{Z},2}\left(\boldsymbol{x}^{\mathrm{Z},2}-\boldsymbol{x}^{\mathrm{Z},2*}\right)\right\|_2^2 \\
&= \left\|\begin{pmatrix} \boldsymbol{A}^{\mathrm{Z}} & \boldsymbol{a} & -\boldsymbol{a} \\ \boldsymbol{0} & w & -w \end{pmatrix}\begin{pmatrix} \boldsymbol{x}-\boldsymbol{s}^* \\ \beta-\alpha \\ \alpha-\beta \end{pmatrix}\right\|_2^2 \\
&= \left\|\begin{pmatrix} \boldsymbol{A}^{\mathrm{Z}}\left(\boldsymbol{x}-\boldsymbol{s}^*\right)+2\left(\beta-\alpha\right)\boldsymbol{a} \\ 2w\left(\alpha-\beta\right) \end{pmatrix}\right\|_2^2 \\
&= \left\|\boldsymbol{A}\left(\boldsymbol{x}-\boldsymbol{s}^*\right)+2\left(\beta-\alpha\right)\boldsymbol{a}\right\|_2^2+4w^2\left(\alpha-\beta\right)^2.
\end{aligned}$$

Since $\left\|\boldsymbol{A}^{\mathrm{Z},2}\boldsymbol{x}^{\mathrm{Z},2}-\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z},2}}\boldsymbol{c}^{\mathrm{Z},2}\right\|_2 \leq \delta$, we have

$$4w^2\left(\alpha-\beta\right)^2 \leq \delta^2,$$

that is,

$$\left(\alpha-\beta\right)^2 \leq \frac{\delta^2}{4w^2}. \tag{3.38}$$

Similarly, we have

$$\left\|\boldsymbol{A}^{\mathrm{Z}}\left(\boldsymbol{x}-\boldsymbol{s}^*\right)+2\left(\beta-\alpha\right)\boldsymbol{a}\right\|_2 \leq \delta.$$

By the triangle inequality,

$$\left\|\boldsymbol{A}^{\mathrm{Z}}\left(\boldsymbol{x}-\boldsymbol{s}^*\right)\right\|_2-2\left\|\left(\beta-\alpha\right)\boldsymbol{a}\right\|_2 \leq \delta.$$

Plugging Equation (3.38) and (3.37) into the above inequality, and rearranging it,

$$\left\|\boldsymbol{A}^{\mathrm{Z}}\left(\boldsymbol{x}-\boldsymbol{s}^*\right)\right\|_2 \leq \left(1+\frac{\|\boldsymbol{a}\|_2}{w}\right)\epsilon_2\left\|\boldsymbol{c}^{\mathrm{Z},2}\right\|_2.$$

By Lemma 2.2.5,

$$\left\|\boldsymbol{A}^{\mathrm{Z}}\left(\boldsymbol{x}-\boldsymbol{s}^{*}\right)\right\|_{2} \leq\left(1+\frac{\|\boldsymbol{a}\|_{2}}{w}\right) \epsilon_{2} \sigma_{\max }\left(\boldsymbol{A}^{\mathrm{Z}}\right)\left\|\boldsymbol{c}^{\mathrm{Z}, 2}\right\|_{2}\left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}} \boldsymbol{c}^{\mathrm{Z}}\right\|_{2}.$$

Together with Lemma 3.6.12 and Equation (3.36), we have

$$\left\|\boldsymbol{A}^{\mathrm{Z}}(\boldsymbol{x}-\boldsymbol{x}^{*})\right\|_{2} \leq\left(\frac{\|\boldsymbol{a}\|_{2}}{w}+\left(1+\frac{\|\boldsymbol{a}\|_{2}}{w}\right) \epsilon_{2}\right) \sigma_{\max }\left(\boldsymbol{A}^{\mathrm{Z}}\right)\left\|\boldsymbol{c}^{\mathrm{Z}}\right\|_{2}\left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}} \boldsymbol{c}^{\mathrm{Z}}\right\|_{2}.$$

According to our setting of w and $\boldsymbol{a}$ in line 1 and line 5 of Algorithm 10,

$$w \geq \frac{\|\boldsymbol{a}\|_{2}}{\epsilon_{2}}.$$

This implies that

$$\left\|\boldsymbol{A}^{\mathrm{Z}} \boldsymbol{x}-\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}} \boldsymbol{c}^{\mathrm{Z}}\right\|_{2} \leq \epsilon_{1}\left\|\boldsymbol{\Pi}_{\boldsymbol{A}^{\mathrm{Z}}} \boldsymbol{c}^{\mathrm{Z}}\right\|_{2},$$

which completes the proof. □

We then bound the singular values of $\boldsymbol{A}^{\mathrm{Z},2}$.

Claim 3.6.14. $\lambda_{\max }\left((\boldsymbol{A}^{\mathrm{Z},2})^{\top} \boldsymbol{A}^{\mathrm{Z},2}\right) \leq O\left(\epsilon_{1}^{-2} m \lambda_{\max }((\boldsymbol{A}^{\mathrm{Z}})^{\top} \boldsymbol{A}^{\mathrm{Z}})\left\|\boldsymbol{A}^{\mathrm{Z}}\right\|_{\infty}^{2}\left\|\boldsymbol{c}^{\mathrm{Z}}\right\|_{2}^{2}\right)$.

Proof. Let λ_1 bet he largest eigenvalue of $(\boldsymbol{A}^{\mathrm{Z},2})^{\top} \boldsymbol{A}^{\mathrm{Z},2}$, and $\boldsymbol{y}=(\tilde{\boldsymbol{y}} ; \alpha ;-\alpha)$ be the associated eigenvector of unit length. Let μ_1 be the largest eigenvalue of $(\boldsymbol{A}^{\mathrm{Z}})^{\top} \boldsymbol{A}^{\mathrm{Z}}$.

$$\begin{aligned}
\lambda_{1} &=\left\|\begin{pmatrix} \boldsymbol{A}^{\mathrm{Z}} & \boldsymbol{a} & -\boldsymbol{a} \\ \mathbf{0} & w & -w \end{pmatrix} \boldsymbol{y}\right\|_{2}^{2} \\
&=\left\|\begin{pmatrix} \boldsymbol{A}^{\mathrm{Z}} & \boldsymbol{a} & -\boldsymbol{a} \\ \mathbf{0} & 1 & -1 \end{pmatrix} \boldsymbol{y}+\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & w-1 & -w+1 \end{pmatrix} \boldsymbol{y}\right\|_{2}^{2} \\
&=\tilde{\boldsymbol{y}}^{\top}(\boldsymbol{A}^{\mathrm{Z}})^{\top} \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}}+4 \alpha \boldsymbol{a}^{\top} \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}}+4 \alpha^{2}\left(\boldsymbol{a}^{\top} \boldsymbol{a}+1\right)+8 \alpha^{2}(w-1)+4 \alpha^{2}(w-1)^{2}.
\end{aligned}$$

By the Courant-Fischer Theorem,

$$\tilde{\boldsymbol{y}}^\top (\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}} \leq \mu_1 \tilde{\boldsymbol{y}}^\top \tilde{\boldsymbol{y}} \leq \mu_1.$$

By Cauchy-Schwarz inequality,

$$\left|\boldsymbol{a}^\top \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}}\right| \leq \|\boldsymbol{a}\|_2 \left\|\boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}}\right\|_2 \leq \|\boldsymbol{A}\|_\infty \sqrt{m}\sqrt{\mu_1}.$$

Thus,

$$\begin{aligned}
\lambda_1 &\leq \mu_1 + 4\left|\alpha\right| \left\|\boldsymbol{A}^{\mathrm{Z}}\right\|_\infty \sqrt{m}\sqrt{\mu_1} + 4\alpha^2 \left(\left\|\boldsymbol{A}^{\mathrm{Z}}\right\|_\infty^2 m + 1\right) + 12\alpha^2 w^2 \\
&\leq \left(\sqrt{\mu_1} + 2\left\|\boldsymbol{A}^{\mathrm{Z}}\right\|_\infty \sqrt{m}\right)^2 + 12\alpha^2 w^2 \\
&\leq 2\mu_1 + 8m \left\|\boldsymbol{A}^{\mathrm{Z}}\right\|_\infty^2 + 12\alpha^2 w^2.
\end{aligned}$$

Our setting of w in line 1 of Algorithm 10 gives

$$\lambda_{\max}\left((\boldsymbol{A}^{\mathrm{Z},2})^\top \boldsymbol{A}^{\mathrm{Z},2}\right) \leq 2\mu_1 + 8m \left\|\boldsymbol{A}^{\mathrm{Z}}\right\|_\infty^2 + 108\epsilon_1^{-2} m\mu_1 \left\|\boldsymbol{A}^{\mathrm{Z}}\right\|_\infty^2 \left\|\boldsymbol{c}^{\mathrm{Z}}\right\|_2^2.$$

This completes the proof. □

Claim 3.6.15. $\lambda_{\min}((\boldsymbol{A}^{\mathrm{Z},2})^\top \boldsymbol{A}^{\mathrm{Z},2}) \geq \frac{2\lambda_{\min}((\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}})}{2\left(\left\|\boldsymbol{A}^{\mathrm{Z}}\right\|_\infty^2 m+1\right)+\lambda_{\min}((\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}})}$.

Proof. Let

$$\boldsymbol{C} = \begin{pmatrix} \boldsymbol{A}^{\mathrm{Z}} & \boldsymbol{a} & -\boldsymbol{a} \\ \boldsymbol{0} & 1 & -1 \end{pmatrix}.$$

Note

$$\boldsymbol{A}^{\mathrm{Z},2} = \boldsymbol{C} + \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & w-1 & -w+1 \end{pmatrix}.$$

We can check that

$$(\boldsymbol{A}^{\mathrm{Z},2})^\top \boldsymbol{A}^{\mathrm{Z},2} = \boldsymbol{C}^\top \boldsymbol{C} + \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & w^2-1 & -(w^2-1) \\ \mathbf{0} & -(w^2-1) & w^2-1 \end{pmatrix}.$$

By our setting, $w^2 - 1 \geq 0$. The second matrix is a rank-one PSD matrix. Since $\boldsymbol{A}^{\mathrm{Z},2}$ and $\boldsymbol{C}$ has same null space,

$$\lambda_{\min}\left((\boldsymbol{A}^{\mathrm{Z},2})^\top \boldsymbol{A}^{\mathrm{Z},2}\right) \geq \lambda_{\min}\left(\boldsymbol{C}^\top \boldsymbol{C}\right).$$

It suffices to lower bound the smallest nonzero eigenvalue of $\boldsymbol{C}^\top \boldsymbol{C}$.

Let $\lambda_k \stackrel{\text{def}}{=} \lambda_{\min}\left(\boldsymbol{C}^\top \boldsymbol{C}\right)$, and $\boldsymbol{y} = (\tilde{\boldsymbol{y}}; \alpha; -\alpha)$ be the associated eigenvector of unit length. Let $\mu_k \stackrel{\text{def}}{=} \lambda_{\min}\left((\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}}\right)$, and $\boldsymbol{x}$ be the associated eigenvector of unit length.

$$\begin{aligned} \lambda_k &= \tilde{\boldsymbol{y}}^\top (\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}} + 4\alpha \boldsymbol{a}^\top \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}} + 4\alpha^2 \left(\boldsymbol{a}^\top \boldsymbol{a} + 1\right) \\ &= \left\| \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}} + 2\alpha \boldsymbol{a} \right\|_2^2 + 4\alpha^2 \\ &\geq 4\alpha^2. \end{aligned} \tag{3.39}$$

On the other hand,

$$\begin{aligned} \lambda_k &= \left(2\alpha\sqrt{\boldsymbol{a}^\top \boldsymbol{a} + 1} + \frac{\boldsymbol{a}^\top \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}}}{\sqrt{\boldsymbol{a}^\top \boldsymbol{a} + 1}}\right)^2 + \tilde{\boldsymbol{y}}^\top (\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}} - \frac{\tilde{\boldsymbol{y}}^\top (\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{a}\boldsymbol{a}^\top \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}}}{\boldsymbol{a}^\top \boldsymbol{a} + 1} \\ &\geq \tilde{\boldsymbol{y}}^\top (\boldsymbol{A}^{\mathrm{Z}})^\top \left(\boldsymbol{I} - \frac{\boldsymbol{a}\boldsymbol{a}^\top}{\boldsymbol{a}^\top \boldsymbol{a} + 1}\right) \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}}. \end{aligned}$$

Take eigen-decomposition of the matrix in the middle,

$$\lambda_k \geq \tilde{\boldsymbol{y}}^\top (\boldsymbol{A}^{\mathrm{Z}})^\top \boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^\top \boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}},$$

where $\boldsymbol{D} := \textsc{diag}\left(1 - \frac{\boldsymbol{a}^\top \boldsymbol{a}}{\boldsymbol{a}^\top \boldsymbol{a}+1}, 1, \ldots, 1\right)$. By Claim 3.6.10, $\boldsymbol{y} \perp \mathrm{null}(\boldsymbol{A}^{\mathrm{Z},2})$ implies $\tilde{\boldsymbol{y}} \perp$

$\mathrm{null}(\boldsymbol{A}^{\mathrm{Z}})$. By the Courant-Fischer Theorem,

$$\lambda_k \geq \frac{1}{\boldsymbol{a}^\top \boldsymbol{a}+1} \left\|\boldsymbol{A}^{\mathrm{Z}} \tilde{\boldsymbol{y}}\right\|_2^2 \geq \frac{1}{\boldsymbol{a}^\top \boldsymbol{a}+1} \mu_k \left\|\tilde{\boldsymbol{y}}\right\|_2^2.$$

Together with Equation (3.39),

$$\lambda_k \geq \max\left\{4\alpha^2, \frac{\mu_k}{\boldsymbol{a}^\top \boldsymbol{a}+1} \left\|\tilde{\boldsymbol{y}}\right\|_2^2\right\}.$$

Since $\|\boldsymbol{y}\|_2^2 = \tilde{\boldsymbol{y}}_2^2 + 2\alpha^2 = 1$,

$$\lambda_k \geq \frac{2\mu_k}{2\left(\boldsymbol{a}^\top \boldsymbol{a}+1\right)+\mu_k} \geq \frac{2\mu_k}{2\left(\|\boldsymbol{A}\|_\infty^2 m+1\right)+\mu_k}.$$

This completes the proof. □

The above two lemmas give the following bound on the condition number of $\boldsymbol{A}^{\mathrm{Z},2}$.

Lemma 3.6.16. $\kappa(\boldsymbol{A}^{\mathrm{Z},2}) = O\left(\epsilon_1^{-1}\sqrt{m}\left\|\boldsymbol{A}^{\mathrm{Z}}\right\|_\infty \left\|\boldsymbol{c}^{\mathrm{Z}}\right\|_2 \left(\sigma_{\max}(\boldsymbol{A}^{\mathrm{Z}}) + \kappa(\boldsymbol{A}^{\mathrm{Z}})\sqrt{m}\left\|\boldsymbol{A}^{\mathrm{Z}}\right\|_\infty\right)\right)$.

Proof of Lemma 3.2.4. By Lemma 3.6.2 and 3.6.9, we have

$$\mathrm{nnz}(\boldsymbol{A}^{\mathrm{Z},2}) = O(s).$$

The smallest nonzero entry does not change. The largest entry of $\boldsymbol{A}^{\mathrm{Z},2}$ is at most

$$\max\{O(\|\boldsymbol{A}\|_\infty), w\} = O\left(\epsilon^{-1} n^2 \sqrt{m}\sigma_{\max}(\boldsymbol{A}) \|\boldsymbol{A}\|_\infty \left\|\boldsymbol{c}^{\mathrm{A}}\right\|_2\right).$$

By Claim 3.1.2 and 3.1.2, the largest entry is upper bounded by

$$O\left(\epsilon^{-1} s^{9/2} U^3\right).$$

By Lemma 3.6.8 and 3.6.16, and Claim 3.1.2 and 3.1.2

$$\kappa(\boldsymbol{A}^{\mathrm{Z},2}) = O\left(\epsilon^{-1} s^8 U^3 K\right).$$

By Lemma 3.6.4 and 3.6.13, we have

$$\epsilon_2^{-1} = O\left(s^{5/2} U^2\right) \epsilon^{-1}.$$

This completes the proof. □

3.7 2D Trusses

In this section, we show that the matrix $\boldsymbol{B}$ constructed by REDUCE $\mathcal{G}_{\mathrm{z},2}TO\mathcal{MC}_2$ (Algorithm 1), is a 2D Truss Incidence Matrix defined in Definition 2.3.3. It follows that for any function f, $\mathcal{G} \leq_f \mathcal{MC}_2$ implies $\mathcal{G} \leq_f \mathcal{T}_2$.

We assume that the algorithm REDUCE $\mathcal{G}_{\mathrm{z},2}TO\mathcal{MC}_2$ is called on a matrix $\boldsymbol{A}$ with no two identical rows: if there are identical rows, these rows can be collapsed into one row without changing the associated normal equations, by reweighting the resulting row, similar to the technique used in the proof of Claim 3.8.4. Details are left to the reader. A key step is to show that a 2-commodity gadget in the reduction corresponds to a 2D truss subgraph, which we call the 2D-truss gadget.

Without loss of generality, we let $\boldsymbol{u}$-variables correspond to the horizonal axis and $\boldsymbol{v}$-variables to the vertical axis of the 2D plane. According to Definition 2.3.1 and 2.3.3:

1. an equation $\boldsymbol{u}_i - \boldsymbol{u}_j = 0$ in a 2-commodity linear system corresponds to a horizontal edge in the 2D plane;

2. an equation $\boldsymbol{v}_i - \boldsymbol{v}_j = 0$ in a 2-commodity linear system corresponds to a vertical edge in the 2D plane;

3. an equation $\boldsymbol{u}_i - \boldsymbol{v}_i - (\boldsymbol{u}_j - \boldsymbol{v}_j) = 0$ in a 2-commodity linear system corresponds to a diagonal edge in the 2D plane.

Note that our reduction here heavily relies on the ability to choose arbitrary weights. In particular, the weights on the elements are not related at all with the distances between the corresponding vertices.

Our strategy for picking the coordinates of the vertices of the constructed 2D truss is the following: we first pick the coordinates of the original n vertices randomly, and then determine the coordinates of the new vertices constructed in the reduction to satisfy all the truss equations.

For the n original vertices, we pick their $\boldsymbol{u}$-coordinates arbitrarily and pick their $\boldsymbol{v}$-coordinates randomly. We pick an n-dimensional random vector $\boldsymbol{y}$ uniformly distributed on the n-dimensional sphere centered at the origin and with radius $R = \|\boldsymbol{A}\|_1 n^{10}$. We then round each entry of $\boldsymbol{y}$ to have precision $\delta = 10^{-10}$, so that the total number of bits used to store an entry is at most $O(\log(n \|\boldsymbol{A}\|_1))$. Let $\tilde{\boldsymbol{y}}$ be the vector after rounding. We assign the $\boldsymbol{v}$-coordinate of the ith vertex to be the ith entry of $\tilde{\boldsymbol{y}}$.

We then pick the coordinates of the new vertices in the order they are created. Note that each time we replace two vertices in the current equations, say $\boldsymbol{s}^{j_1}, \boldsymbol{s}^{j_2}$, whose coordinates have already been determined, we create a 2D truss gadget with 7 new vertices, say $\boldsymbol{s}^t, \boldsymbol{s}^{t+1}, \ldots, \boldsymbol{s}^{t+6}$ (See Algorithm 2 $\mathcal{MC}_2$GADGET for the construction.). According to the construction of this gadget, the new vertices $\boldsymbol{s}^{t+1}, \ldots, \boldsymbol{s}^{t+6}$ only appear in this single gadget, whose coordinates do not affect other vertices. Figure 3.2 is the corresponding subgraph which satisfies all the equations in the 2D truss gadget. Note the two triangles $(\boldsymbol{s}^{t+3}, \boldsymbol{s}^{t+5}, \boldsymbol{s}^{t+6})$ and $(\boldsymbol{s}^{t+3}, \boldsymbol{s}^{t+4}, \boldsymbol{s}^{t+5})$ need to be isosceles right triangles, which implies $\boldsymbol{v}_t = (\boldsymbol{v}_{j_1} + \boldsymbol{v}_{j_2})/2$. Note also that we can assign $\boldsymbol{u}$-coordinates to the new vertices which are not between the $\boldsymbol{u}$-coordinates of $\boldsymbol{s}^{j_1}$ and $\boldsymbol{s}^{j_2}$. In fact, using an appropriate choice of $\boldsymbol{u}$-coordinates and edge weights, we can always place $\boldsymbol{s}^t, \boldsymbol{s}^{t+1}, \ldots, \boldsymbol{s}^{t+6}$ to get the desired equations, provided $\boldsymbol{v}_{j_1} \neq \boldsymbol{v}_{j_2}$, which we later will argue holds with high probability using

Lemma 3.7.1. First, however, we state and prove Lemma 3.7.1.

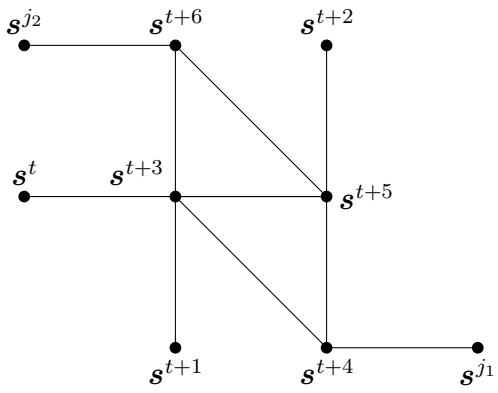

Figure 3.2: Geometric realization of the mutlicommodity flow gadget generated by as a truss matrix

Lemma 3.7.1. *Let $\boldsymbol{a} \in \mathbb{R}^n$ be a fixed vector such that $-2 \leq \boldsymbol{a}_i \leq 2, \forall i \in [n]$ and $\boldsymbol{a}^\top \mathbf{1} = 0$, and $\boldsymbol{a} \neq \mathbf{0}$. Let $\tilde{\boldsymbol{y}}$ be a vector picked as above. Then,*

$$\Pr\left(\boldsymbol{a}^\top \tilde{\boldsymbol{y}} = 0\right) \leq \frac{2\delta n^2}{\|\boldsymbol{a}\|_2 R}.$$

Proof. Let $\Delta \stackrel{\text{def}}{=} \tilde{\boldsymbol{y}} - \boldsymbol{y}$. Clearly, $-\delta \leq \Delta_i \leq \delta, \forall i \in [n]$.

$$\boldsymbol{a}^\top \tilde{\boldsymbol{y}} = \boldsymbol{a}^\top (\boldsymbol{y} + \Delta) \leq \boldsymbol{a}^\top \boldsymbol{y} + \sum_{i \in [n]} |\boldsymbol{a}_i| \, |\Delta_i| \leq \boldsymbol{a}^\top \boldsymbol{y} + 2\delta n.$$

Similarly, $\boldsymbol{a}^\top \tilde{\boldsymbol{y}} \geq \boldsymbol{a}^\top \boldsymbol{y} - 2\delta n$. Thus,

$$\Pr\left(\boldsymbol{a}^\top \tilde{\boldsymbol{y}} = 0\right) \leq \Pr\left(\left|\boldsymbol{a}^\top \boldsymbol{y}\right| \leq 2\delta n\right).$$

Since the distribution of $\boldsymbol{y}$ is rotation invariant, we assume without loss of generality $\boldsymbol{a} =$

$(\|\boldsymbol{a}\|_2, 0, \ldots, 0)$. Let A be the area of the n-dimensional sphere.

$$A = \frac{2\pi^{(n+1)/2}}{\Gamma\left(\frac{n+1}{2}\right)} \cdot R^n.$$

Let A_δ be the area of $\{\boldsymbol{y} : \|\boldsymbol{y}\|_2 = R, |\boldsymbol{y}_1| \leq 2\delta n / \|\boldsymbol{a}\|_2\}$. Then,

$$A_\delta \leq \frac{2\pi^{n/2}}{\Gamma\left(\frac{n}{2}\right)} \cdot R^{n-1} \cdot \frac{2\delta n}{\|\boldsymbol{a}\|_2}.$$

Thus, (assume n is even, the case of odd n can be checked similarly)

$$\begin{aligned}
\Pr\left(\left|\boldsymbol{a}^\top \boldsymbol{y}\right| \leq \delta\right) &= \frac{A_\delta}{A} \\
&\leq \frac{2\delta n}{\|\boldsymbol{a}\|_2 \sqrt{\pi} R} \cdot \frac{\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n}{2})} \\
&= \frac{2\delta n}{\|\boldsymbol{a}\|_2 R} \cdot \frac{n!\sqrt{\pi}}{n!!(n-2)!!} \\
&\leq \frac{2\delta n^2}{\|\boldsymbol{a}\|_2 R}.
\end{aligned}$$

This completes the proof. □

By our construction of the truss, for each vertex, its $\boldsymbol{v}$-coordinate can be written as a fixed convex combination of $\tilde{\boldsymbol{y}}$, say $\boldsymbol{c}^\top \tilde{\boldsymbol{y}}$ in which $\boldsymbol{c}^\top \mathbf{1} = 1$ and $\boldsymbol{c}_i \geq 0, \forall i \in [n]$. Note that when algorithm REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{MC}_2$ is applied to a matrix $\boldsymbol{A}$, it processes the jth row in k iterations where by Lemma 3.3.1 we have $k \leq \log \|\boldsymbol{A}_j\|_1$. Let $p \stackrel{\text{def}}{=} \log \|\boldsymbol{A}_j\|_1$. Given how the convex combination specified by $\boldsymbol{c}$ is formed, it follows that $2^p \boldsymbol{c}$ is an integer vector.

Next we argue that when two variables are chosen for pairing by the REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{MC}_2$ algorithm as it processes some row $\boldsymbol{A}_j$, these two variables will have their $\boldsymbol{v}$-coordinates represented as convex combinations $\boldsymbol{c}^\top \tilde{\boldsymbol{y}}$ and $\boldsymbol{d}^\top \tilde{\boldsymbol{y}}$ where crucially $\boldsymbol{c} \neq \boldsymbol{d}$. This ensures that $2^p(\boldsymbol{c} - \boldsymbol{d})$ is a non-zero integer vector, and hence $\|\boldsymbol{c} - \boldsymbol{d}\|_2 \geq 2^{-p} = 1/\|\boldsymbol{A}_j\|_1$. This follows from stronger claim stated below, which we prove later.

Claim 3.7.2. *Suppose algorithm* REDUCE $\mathcal{G}_{z,2}$TO$\mathcal{MC}_2$ *is processing some row* $\boldsymbol{A}_j$*. Let* V^l *be the set of variables with non-zero coefficients in the main equation* $\mathcal{A}_j$ *at the* l*th iteration of the while-loop in Line 20 of* REDUCE $\mathcal{G}_{z,2}$TO$\mathcal{MC}_2$*. Let* S^l *be the set of associated vertices. Consider two arbitrary vertices* $\boldsymbol{s}, \boldsymbol{t} \in S^l$*, and let* $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^n$ *be the non-negative vectors with* $\boldsymbol{c}^\top \mathbf{1} = \boldsymbol{d}^\top \mathbf{1} = 1$ *s.t. the* $\boldsymbol{v}$*-coordinates of* $\boldsymbol{s}$ *and* $\boldsymbol{t}$ *represented as convex combinations are* $\boldsymbol{c}^\top \tilde{\boldsymbol{y}}$ *and* $\boldsymbol{d}^\top \tilde{\boldsymbol{y}}$ *respectively. For every* $i \in [n]$*, we view entries* $\boldsymbol{c}_i$ *and* $\boldsymbol{d}_i$ *as fixed point binary numbers:* $\boldsymbol{c}_i = \alpha_0^i.\alpha_1^i\alpha_2^i \ldots \alpha_p^i$ *and* $\boldsymbol{d}_i = \beta_0^i.\beta_1^i\beta_2^i \ldots \beta_p^i$*, or equivalently* $\boldsymbol{c}_i = \sum_{k=0}^p \alpha_k^i 2^{-k}$ *and* $\boldsymbol{d}_i = \sum_{k=0}^p \beta_k^i 2^{-k}$*. Then there is no index* k *s.t.* $1 = \alpha_k^i = \beta_k^i$*, i.e. there is no index where the* k*th bit is 1 in both strings.*

These two vertices have same $\boldsymbol{v}$-coordinate if and only if $(\boldsymbol{c} - \boldsymbol{d})^\top \tilde{\boldsymbol{y}} = 0$. Let $\boldsymbol{a} \stackrel{\text{def}}{=} \boldsymbol{c} - \boldsymbol{d}$. Then, $-2 \leq \boldsymbol{a}_i \leq 2, \forall i \in [n]$, $\boldsymbol{a}^\top \mathbf{1} = 0$, and

$$\|\boldsymbol{a}\|_2 \geq 1/\|\boldsymbol{A}_j\|_1 .$$

By Lemma 3.7.1,

$$\Pr\left(\boldsymbol{c}^\top \tilde{\boldsymbol{y}} = \boldsymbol{d}^\top \tilde{\boldsymbol{y}}\right) \leq \frac{2\delta n^2 \|\boldsymbol{A}_j\|_1}{R}.$$

By Lemma 3.3.1, the total number of the vertices in the truss is at most

$$O\left(n^2 \log n\right).$$

By a union bound, the probability that there exist two different vertices with same $\boldsymbol{v}$-coordinate is at most

$$\frac{2\delta n^2 \|\boldsymbol{A}\|_1}{R} \cdot O\left(n^2 \log n\right)^2 = O\left(\frac{\log^2 n}{n^4}\right).$$

Proof of Lemma 3.2.8. Since the linear system for 2D trusses is the same as the linear system for 2-commodity, all complexity parameters of these two linear systems are the same. □

Proof of Claim 3.7.2. We prove the claim by induction. Our induction hypothesis is simply that the claim holds in round l.

Note that by Lemma 3.3.1 every time a new vertex is created (during the lth iteration of while-loop in Line 20), it is always paired in the following iteration (iteration $l+1$) and then disappears (i.e. has zero coefficient) in the main equation in all following iterations.

Observe that the convex combination vector $\boldsymbol{a}$ for each vertex that corresponds to an original variable i is has $\boldsymbol{a}_i = 1$ and $\boldsymbol{a}_h = 0$ for all $h \neq i$. This proves the induction hypothesis for the base case of the variables in the main equation $\mathcal{A}_j$ before the first iteration of the while-loop (i.e. $l = 0$).

Suppose $\boldsymbol{c}$,$\boldsymbol{d}$ are the convex combination vectors for two variables that exist in some round l. Assume the induction hypothesis holds for round $l-1$. Write the binary strings for the ith of both vectors as $\boldsymbol{c}_i = \alpha_0^i.\alpha_1^i\alpha_2^i\ldots\alpha_p^i$ and $\boldsymbol{d}_i = \beta_0^i.\beta_1^i\beta_2^i\ldots\beta_p^i$, or equivalently $\boldsymbol{c}_i = \sum_{k=0}^{p}\alpha_k^i 2^{-k}$ and $\boldsymbol{d}_i = \sum_{k=0}^{p}\beta_k^i 2^{-k}$. Suppose for the sake of contradiction that there exists some k s.t. $\alpha_k^i = \beta_k^i = 1$. Trivially, it cannot be the case that the such a collision occurs if either variable is an original variable. So both variables must be new variables. The bit string for entry $\boldsymbol{c}_i$ is created by averaging two bit strings of variables from the main equation in round $l-1$, say $\eta_0^i.\eta_1^i\eta_2^i\ldots\eta_p^i$ and $\gamma_0^i.\gamma_1^i\gamma_2^i\ldots\gamma_p^i$. Similarly, bit string for entry $\boldsymbol{d}_i$ is created by averaging two bit strings of variables from the main equation in round $l-1$, say $\theta_0^i.\theta_1^i\theta_2^i\ldots\theta_p^i$ and $\sigma_0^i.\sigma_1^i\sigma_2^i\ldots\sigma_p^i$. Note that each variable can only be paired once in each iteration, so the four bit strings must come from distinct variables in round l.

$\alpha_k^i = 1$ requires that exactly one of the following conditions is true:

1. $\gamma_{k-1}^i = 1$
2. $\eta_{k-1}^i = 1$
3. A "carry" occurred when adding strings $\gamma_k^i\gamma_{k+1}^i\ldots\gamma_p^i$ and $\eta_k^i\eta_{k+1}^i\ldots\eta_p^i$.

But, Case 3 immediately leads to a contradiction, as a carry can only occur when there exists some bit position g s.t. $\gamma_g^i = \eta_g^i$. But this is false by the induction hypothesis. Thus

we must be in either Case 1 or Case 2. By similar logic, we can conclude from $\beta_k^i = 1$ that exactly one of the following must be true: either $\theta_{k-1}^i = 1$ or $\sigma_{k-1}^i = 1$. All together, we have concluded that exactly two of the four bits $\eta_{k-1}^i, \gamma_{k-1}^i, \theta_{k-1}^i$, and σ_{k-1}^i must be set to 1. This contradicts the induction hypothesis. Having established a contradiction whenever the induction hypothesis fails at step l, we have shown that it holds at this step. □

3.8 Connections with Interior Point Methods

In this section, we discuss how applications in scientific computing and combinatorial optimization produce the linear systems that we show are hard to solve. We first give a brief overview of interior point methods, with focus on how they generate linear systems in Section 3.8.1. Then we formalize the matrices that interior point methods produce when run on 2-commodity flow matrices in Section 3.8.2 and isotropic total variation minimization in Section 3.8.3.

3.8.1 Brief Overview of Interior Point Methods

Interior point methods [70, 71, 72, 73, 24, 4, 74] can be viewed as ways of solving convex optimization problems via a sequence of linear systems. For simplicity, we choose the log-barrier based interpretation from Chapter 11 of the book by Boyd and Vandenberghe as our starting point. The main idea is to represent a convex optimization problem with constraints as a sequence of linear programs with terms called barrier functions added to the objective. Specifically, turning the problem:

$$\begin{aligned} \min_{\boldsymbol{y}} \quad & f(\boldsymbol{y}) \\ \text{subject to:} \quad & \boldsymbol{M}\boldsymbol{y} = \boldsymbol{b} \\ & \boldsymbol{y} \geq 0 \end{aligned}$$

into the equality-constrained optimization problem:

$$\begin{aligned} \min_{\boldsymbol{y}} \quad & t \cdot f(\boldsymbol{y}) - \sum_i \log(\boldsymbol{y}_i) \\ \text{subject to:} \quad & \boldsymbol{M}\boldsymbol{y} = \boldsymbol{b} \end{aligned}$$

for a parameter t that is gradually increased (by factors of about $1 + n^{-1/2}$ throughout the course of the algorithm). The solution of the above optimization problem converges to an optimal solution of the original linear programming, as t goes to infinity. Between these increase steps, the algorithm performs Newton steps on this log barrier objective, which when combined with the equality constraint $\boldsymbol{M}\boldsymbol{y} = \boldsymbol{b}$ requires solving the problem:

$$\begin{aligned} \min_{\Delta \boldsymbol{y}} \quad & -\boldsymbol{g}(\boldsymbol{y})^\top \Delta \boldsymbol{y} \\ \text{subject to:} \quad & \boldsymbol{M}\Delta\boldsymbol{y} = \boldsymbol{0} \\ & \|\Delta \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{y})} \leq 0.1 \end{aligned}$$

which is to maximize the projection along the gradient subject to the second order term being at most 0.1 and staying in the null space. This can in turn be interpreted as a least squares problem, and solving the linear system:

$$\boldsymbol{M}\boldsymbol{H}(\boldsymbol{y})^{-1}\boldsymbol{M}^\top \boldsymbol{x} = -t\boldsymbol{M}\boldsymbol{H}(\boldsymbol{y})^{-1}\boldsymbol{g}(\boldsymbol{y}). \tag{3.40}$$

3.8.2 2-Commodity Flow

We now show that solving 2-commodity flow problems using interior point methods as described in Subsection 3.8.1 can lead to any system in the class $\mathcal{MC}_2^{\geq 0}$. There are also many variants of the multicommodity flow problem [75], and we work with the minimum cost version due to it being the most general.

Definition 3.8.1 (Min-cost 2-commodity flow problem)**.** *Given a directed graph* $G =$

(V, E) with n vertices and m edges, a positive edge-capacity vector $\boldsymbol{z} \in \mathbb{R}^m$, a positive edge-cost vector $\boldsymbol{c} \in \mathbb{R}^{2m}$, and two vertex-demand vectors $\boldsymbol{d}^1, \boldsymbol{d}^2 \in \mathbb{R}^n$. The goal is to compute two flows $\boldsymbol{y}^1, \boldsymbol{y}^2$ such that,

1. *$\boldsymbol{y}^1$ satisfies the demand $\boldsymbol{d}^1$, and $\boldsymbol{y}^2$ satisfies the demand $\boldsymbol{d}^2$,*
2. *for each edge $e \in E$, the sum of the two flows on e is no larger than the edge capacity $\boldsymbol{z}_e$, and*
3. *the total cost of the two flows is minimized.*

To formulate this as a linear program, we let $\boldsymbol{y} = (\boldsymbol{y}^1; \boldsymbol{y}^2)$ be the two flows, and $\boldsymbol{N}$ be the edge-vertex incidence matrix of graph G.

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{y} \\ s.t. \quad & \boldsymbol{N}^\top \boldsymbol{y}^1 = \boldsymbol{d}^1 \\ & \boldsymbol{N}^\top \boldsymbol{y}^2 = \boldsymbol{d}^2 \\ & \boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{z} - \boldsymbol{y}^1 - \boldsymbol{y}^2 \geq \boldsymbol{0} \end{aligned}$$

Write this linear programming as the following minimization problem with a logarithmic barrier function,

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{y} - \frac{1}{t} \sum_{i \in [m]} \log \boldsymbol{y}_i^1 + \log \boldsymbol{y}_i^2 + \log(\boldsymbol{z}_i - \boldsymbol{y}_i^1 - \boldsymbol{y}_i^2) \\ s.t. \quad & \boldsymbol{N}^\top \boldsymbol{y}^1 = \boldsymbol{d}^1 \\ & \boldsymbol{N}^\top \boldsymbol{y}^2 = \boldsymbol{d}^2 \end{aligned}$$

where $t > 0$ is a parameter.

Definition 3.8.2. *We say a linear system $\boldsymbol{B}^\top \boldsymbol{B} \boldsymbol{x} = \boldsymbol{B}^\top \boldsymbol{c}$ is a* Minimum Cost 2-commodity Flow IPM Linear System *if it can be obtained from as an instance of the Newton-Step Linear System of Equation* (3.40) *for some Min-cost 2-commodity flow problem.*

The class of Minimum Cost 2-commodity Flow IPM Linear Systems is appears more restrictive than $\mathcal{MC}_2^{>0}$, but as we will see, it is essentially equivalent, and still sufficiently expressive that any linear system can be reduced to it. The main result that we will sketch in this section is:

Lemma 3.8.3. *For any linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}$ (with error parameter ϵ) with polynomially bounded sparse parameter complexity, there exists an efficient reduction to a Minimum Cost 2-commodity Flow IPM Linear System.*

Our reduction from a general linear system $(\boldsymbol{A}, \boldsymbol{c}, \epsilon)$, where $\boldsymbol{A} \in \mathcal{G}$, to a Minimum Cost 2-commodity Flow IPM Linear System is the same as our reduction to $\mathcal{MC}_2^{>0}$, except that before our chain of reductions, we first multiply the linear system by a diagonal matrix $\boldsymbol{S}$ with diagonal entries that are ± 1. We will later specify how to choose these signs. This gives us the linear system $\boldsymbol{S}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{S}\boldsymbol{c}$ with the same error parameter as before. It is easy to verify that this system has the same sparse parameter complexity as the original linear system, and an approximate solution to this system is an approximate solution to the original system with the same ϵ.

We now apply our usual chain of reductions to get a linear system in over a matrix in $\mathcal{MC}_2^{>0}$. We write this system as

$$(\boldsymbol{B}^{>0})^\top \boldsymbol{B}^{>0} \boldsymbol{x} = (\boldsymbol{B}^{>0})^\top \boldsymbol{c}^{\mathrm{B}^{>0}}.$$

The remainder of this Section is dedicated to showing that this linear system is a Minimum Cost 2-commodity Flow IPM Linear System, thus proving Lemma 3.8.3.

We can pull out the edge weights from $\boldsymbol{B}^{>0}$ by writing $(\boldsymbol{B}^{>0})^\top \boldsymbol{B}^{>0} = \widehat{\boldsymbol{B}}^\top \boldsymbol{W} \widehat{\boldsymbol{B}}$, where $\widehat{\boldsymbol{B}}$ is the unweighted 2-commodity edge-vertex incidence matrix with the same edge structure as $\boldsymbol{B}^{>0}$, and $\boldsymbol{W}$ is the diagonal matrix of the edge weights. Then, the linear

system in $\mathcal{MC}_2^{>0}$ can be written as

$$\widehat{\boldsymbol{B}}^\top \boldsymbol{W} \widehat{\boldsymbol{B}} \boldsymbol{x} = \widehat{\boldsymbol{B}}^\top \boldsymbol{W}^{1/2} \boldsymbol{c}^{\mathrm{B}^{>0}}. \tag{3.41}$$

Before we prove Lemma 3.8.3, we explore some properties of the above linear system.

Claim 3.8.4. *There exist a 2-commodity edge-vertex incidence matrix $\widetilde{\boldsymbol{B}}$, a diagonal matrix $\widetilde{\boldsymbol{W}}$, and a vector $\tilde{\mathbf{c}}$ such that*

1. $\widetilde{\boldsymbol{B}}^\top \widetilde{\boldsymbol{W}} \widetilde{\boldsymbol{B}} = \widehat{\boldsymbol{B}}^\top \boldsymbol{W} \widehat{\boldsymbol{B}}$ *and* $\widehat{\boldsymbol{B}}^\top \boldsymbol{W}^{1/2} \boldsymbol{c}^{B^{>0}} = \widetilde{\boldsymbol{B}}^\top \boldsymbol{W}^{1/2} \tilde{\mathbf{c}}$,
2. *all the rows of $\widetilde{\boldsymbol{B}}$ are distinct.*

Proof. If all rows of $\widehat{\boldsymbol{B}}$ are distinct, then we set $\widetilde{\boldsymbol{B}} = \widehat{\boldsymbol{B}}$, $\widetilde{\boldsymbol{W}} = \boldsymbol{W}$ and $\tilde{\mathbf{c}} = \boldsymbol{c}^{\mathrm{B}^{>0}}$. Otherwise, assume the ith row and the jth row of $\widehat{\boldsymbol{B}}$ are the same. Note

$$\widehat{\boldsymbol{B}}^\top \boldsymbol{W} \widehat{\boldsymbol{B}} = \sum_{k \neq i,j} \boldsymbol{W}_{kk} \widehat{\boldsymbol{B}}_k^\top \widehat{\boldsymbol{B}}_k + (\boldsymbol{W}_{ii} + \boldsymbol{W}_{jj}) \widehat{\boldsymbol{B}}_i^\top \widehat{\boldsymbol{B}}_i,$$

and

$$\widehat{\boldsymbol{B}}^\top \boldsymbol{W}^{1/2} \boldsymbol{c}^{\mathrm{B}^{>0}} = \sum_{k \neq i,j} \boldsymbol{W}_{kk}^{1/2} \boldsymbol{c}_k^{\mathrm{B}^{>0}} \widehat{\boldsymbol{B}}_k^\top + (\boldsymbol{W}_{ii}^{1/2} \boldsymbol{c}_i^{\mathrm{B}^{>0}} + \boldsymbol{W}_{jj}^{1/2} \boldsymbol{c}_j^{\mathrm{B}^{>0}}) \widehat{\boldsymbol{B}}_i^\top.$$

We can construct $\widetilde{\boldsymbol{B}}$ and $\widetilde{\boldsymbol{W}}$ by removing the jth row of $\widehat{\boldsymbol{B}}$, $\boldsymbol{W}$ and set the corresponding edge weight $\widetilde{\boldsymbol{W}}_{ii}$ to be $\boldsymbol{W}_{ii} + \boldsymbol{W}_{jj}$. We construct $\tilde{\mathbf{c}}$ by removing the jth entry of $\boldsymbol{c}^{\mathrm{B}^{>0}}$ and set $\tilde{\mathbf{c}}_i$ to be $(\boldsymbol{W}_{ii}^{1/2} \boldsymbol{c}_i^{\mathrm{B}^{>0}} + \boldsymbol{W}_{jj}^{1/2} \boldsymbol{c}_j^{\mathrm{B}^{>0}})/(\boldsymbol{W}_{ii} + \boldsymbol{W}_{jj})^{1/2}$. We repeat the above process to merge the identical rows until we get the desired matrices and vector. □

Remark. Let $\boldsymbol{x}^*$ be a minimizer of $\left\| \boldsymbol{W}^{1/2} \widehat{\boldsymbol{B}} \boldsymbol{x} - \boldsymbol{c}^{\mathrm{B}^{>0}} \right\|_2$. We know that $\boldsymbol{x}^*$ is a solution of the normal equations $\widehat{\boldsymbol{B}}^\top \boldsymbol{W} \widehat{\boldsymbol{B}} \boldsymbol{x} = \widehat{\boldsymbol{B}}^\top \boldsymbol{W}^{1/2} \boldsymbol{c}^{\mathrm{B}^{>0}}$. By Claim 3.8.4, $\boldsymbol{x}^*$ is a solution of $\widetilde{\boldsymbol{B}}^\top \widetilde{\boldsymbol{W}} \widetilde{\boldsymbol{B}}^\top \boldsymbol{x} = \widetilde{\boldsymbol{B}}^\top \widetilde{\boldsymbol{W}}^{1/2} \tilde{\mathbf{c}}$, which in turn gives that $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}} \left\| \widetilde{\boldsymbol{W}}^{1/2} \widetilde{\boldsymbol{B}} \boldsymbol{x} - \tilde{\mathbf{c}} \right\|_2$. Let $\boldsymbol{x}$ be an approximate solution such that

$$\left\| \boldsymbol{x} - \boldsymbol{x}^* \right\|_{\widehat{\boldsymbol{B}}^\top \boldsymbol{W} \widehat{\boldsymbol{B}}} \leq \epsilon \left\| \widehat{\boldsymbol{B}}^\top \boldsymbol{W}^{1/2} \boldsymbol{c}^{\mathrm{B}^{>0}} \right\|_{(\widehat{\boldsymbol{B}}^\top \boldsymbol{W} \widehat{\boldsymbol{B}})^\dagger}.$$

By Claim 3.8.4, the above equation is equivalent to

$$\|\boldsymbol{x}-\boldsymbol{x}^*\|_{\widetilde{\boldsymbol{B}}^\top \widetilde{\boldsymbol{W}} \widetilde{\boldsymbol{B}}} \leq \epsilon \left\|\widetilde{\boldsymbol{B}}^\top \widetilde{\boldsymbol{W}}^{1/2} \tilde{\boldsymbol{c}}\right\|_{(\widetilde{\boldsymbol{B}}^\top \widetilde{\boldsymbol{W}} \widetilde{\boldsymbol{B}})^\dagger}.$$

It means that $\boldsymbol{x}$ is an approximate solution of $\min_{\boldsymbol{x}} \left\| \widetilde{\boldsymbol{W}}^{1/2} \widetilde{\boldsymbol{B}} \boldsymbol{x} - \tilde{\boldsymbol{c}} \right\|_2$ with ϵ-accuracy, vice versa. Thus, the two minimization problems before and after merging identical rows are equivalent.

Without the loss of generality, we assume that all the rows of $\widehat{\boldsymbol{B}}$ are distinct. We define *the underlying graph* of $\widehat{\boldsymbol{B}}$ to be the *simple* graph over n vertices where vertices i and j are connected if and only if $\widehat{\boldsymbol{B}}$ has at least one type of edges between i and j. According to the constructions in REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{MC}_2$ (Algorithm 1) and REDUCE$\mathcal{MC}_2$TO$\mathcal{MC}_2^{>0}$ (Algorithm 4), each edge of the underlying graph has exactly 3 types of edges in $\widehat{\boldsymbol{B}}$ (that is, type 1, type 2 and type $1+2$), and the edge weights of $\boldsymbol{B}^{>0}$ are either at least 1 or equal to a tiny number δ assigned in line 1 of Algorithm 4. We call edge weights which are at least 1 as *large* weights.

Claim 3.8.5. *For each edge in the underlying graph of $\widehat{\boldsymbol{B}}$, exactly one of its type 1, type 2, and type $1+2$ edges has large weight.*

Proof. Note that the large-weight edges all appear in $\boldsymbol{B} \in \mathcal{MC}_2$ constructed in Algorithm 1. An edge in $\boldsymbol{B}$ is either a gadget edge in $\mathcal{B}$ or a non-gadget edge in $\mathcal{A}$.

1. By Algorithm 2, all edges in a single $\mathcal{MC}_2$-gadget are distinct.

2. We show that each gadget edge is distinct from all other edges. By Algorithm 2, in a 2-commodity gadget, each "old" vertex is connected to a "new" vertex, and all new vertices except $\boldsymbol{x}_t$ are independent of the new vertices created in all other gadgets. However, in each gadget including $\boldsymbol{x}_t$ (as a new vertex or an odd vertex), $\boldsymbol{x}_t$ is connected to new vertices which are created in that gadget and do not appear in any other edge outside that gadget.

3. It is possible that two identical non-gadget edges exist when the original linear system instance has redundant or inconsistent constraints. But note all non-gadget edges are type 1 edges, it implies that two identical edges correspond to two identical rows in $\widehat{\boldsymbol{B}}$. By Claim 3.8.4, we can always merge such identical rows so that all non-gadget edges are distinct.

This completes the proof. □

Proof of Lemma 3.8.3. Note the matrices $\widehat{\boldsymbol{B}}$, $\boldsymbol{W}$ and the vector $\boldsymbol{c}^{\mathrm{B}>0}$ are given by the reductions from the general linear system instance $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}^{\mathrm{A}}$. The goal of proving Lemma 3.8.3 is to determine the edge-vertex incidence matrix $\boldsymbol{N}$, the cost vector $\boldsymbol{c}$, the demand vectors $\boldsymbol{d}^1, \boldsymbol{d}^2$, the edge capacity vector $\boldsymbol{z}$, and the flows $\boldsymbol{y}^1, \boldsymbol{y}^2$ such that the intermediate linear systems that arise in an interior point method in Equation (3.40) for solving the Min-cost 2-commodity Flow Problem is exactly the above linear system in Equation (3.41).

We first make the coefficient matrices of the two linear systems in Equation (3.40) and (3.41) to be the same, by choosing the matrix $\boldsymbol{N}$ and the vectors $\boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{d}^1, \boldsymbol{d}^2, \boldsymbol{z}$ properly. The coefficient matrix of the equality constraints of the 2-commodity linear programming is

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{N}^\top & \\ & \boldsymbol{N}^\top \end{pmatrix}. \tag{3.42}$$

We choose $\boldsymbol{N}$ to be the edge-vertex incidence matrix of the underlying graph of $\widehat{\boldsymbol{B}}$. By denoting the residue flow amount along an edge as

$$\boldsymbol{y}^r \stackrel{\text{def}}{=} \boldsymbol{z} - \boldsymbol{y}^1 - \boldsymbol{y}^2,$$

we can rewrite the Hessian as:

$$\boldsymbol{H} = \frac{1}{t} \sum_{i \in [m]} \frac{1}{(\boldsymbol{y}_i^1)^2} \boldsymbol{e}_{2i-1} \boldsymbol{e}_{2i-1}^\top + \frac{1}{(\boldsymbol{y}_i^2)^2} \boldsymbol{e}_{2i} \boldsymbol{e}_{2i}^\top + \frac{1}{(\boldsymbol{y}_i^r)^2} \left(\boldsymbol{e}_{2i-1} + \boldsymbol{e}_{2i}\right) \left(\boldsymbol{e}_{2i-1} + \boldsymbol{e}_{2i}\right)^\top,$$

where $\boldsymbol{e}_i \in \mathbb{R}^{2m}$ is the standard basis vector.

We can rearrange the rows and columns of $\boldsymbol{H}$ such that, row $2i-1$ and column $2i-1$ correspond to the $\boldsymbol{y}^1$-flow of the ith edge, and row $2i$ and column $2i$ correspond to the $\boldsymbol{y}^2$-flow of the ith edge. After this rearrangement, $\boldsymbol{H}$ becomes a block diagonal matrix, where each edge i corresponds to a one 2×2 block given by

$$\boldsymbol{H}_{2i-1:2i} \stackrel{\text{def}}{=} \begin{pmatrix} \frac{1}{(\boldsymbol{y}_i^1)^2} + \frac{1}{(\boldsymbol{y}_i^r)^2} & \frac{1}{(\boldsymbol{y}_i^r)^2} \\ \frac{1}{(\boldsymbol{y}_i^r)^2} & \frac{1}{(\boldsymbol{y}_i^2)^2} + \frac{1}{(\boldsymbol{y}_i^r)^2} \end{pmatrix}. \tag{3.43}$$

The block diagonal structure of $\boldsymbol{H}$ then gives:

$$\boldsymbol{H}^{-1} = \text{DIAG}\left(\boldsymbol{H}_{1:2}^{-1}, \ldots, \boldsymbol{H}_{2m-1:2m}^{-1}\right),$$

and for each edge i we have:

$$\boldsymbol{H}_{2i-1:2i}^{-1} = \frac{1}{\alpha_i} \left(\begin{pmatrix} (\boldsymbol{y}_i^1)^2 (\boldsymbol{y}_i^r)^2 & 0 \\ 0 & (\boldsymbol{y}_i^2)^2 (\boldsymbol{y}_i^r)^2 \end{pmatrix} + (\boldsymbol{y}_i^1)^2 (\boldsymbol{y}_i^2)^2 \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \right), \tag{3.44}$$

where $\alpha_i = (\boldsymbol{y}_i^1)^2 + (\boldsymbol{y}_i^2)^2 + (\boldsymbol{y}_i^r)^2$. It means that

1. the type 1 edge has weight $(\boldsymbol{y}_i^1)^2 (\boldsymbol{y}_i^r)^2 / \alpha_i$,
2. the type 2 edge has weight $(\boldsymbol{y}_i^2)^2 (\boldsymbol{y}_i^r)^2 / \alpha_i$, and
3. the type $1+2$ edge has weight $(\boldsymbol{y}_i^1)^2 (\boldsymbol{y}_i^2)^2 / \alpha_i$.

Note these three types of weights are symmetric. Let $w_{(i,1)}, w_{(i,2)}, w_{(i,1+2)}$ be the edge weights of type 1, type 2 and type $1+2$ edge of the ith edge, respectively. These edge

weights are given by the corresponding diagonals of $\boldsymbol{W}$. We set the values of $\boldsymbol{y}_i^1, \boldsymbol{y}_i^2, \boldsymbol{y}_i^r$ such that

$$
\begin{aligned}
(\boldsymbol{y}_i^1)^2(\boldsymbol{y}_i^r)^2/\alpha_i &= w_{(i,1)}, \\
(\boldsymbol{y}_i^2)^2(\boldsymbol{y}_i^r)^2/\alpha_i &= w_{(i,2)}, \\
(\boldsymbol{y}_i^1)^2(\boldsymbol{y}_i^2)^2/\alpha_i &= w_{(i,1+2)}.
\end{aligned}
$$

Solving the above equations gives:

$$
\begin{aligned}
(\boldsymbol{y}_i^1)^2 &= \frac{w_{(i,1)}w_{(i,1+2)}}{w_{(i,2)}} + w_{(i,1)} + w_{(i,1+2)}, \\
(\boldsymbol{y}_i^2)^2 &= \frac{w_{(i,2)}w_{(i,1+2)}}{w_{(i,1)}} + w_{(i,2)} + w_{(i,1+2)}, \\
(\boldsymbol{y}_i^r)^2 &= \frac{w_{(i,1)}w_{(i,2)}}{w_{(i,1+2)}} + w_{(i,1)} + w_{(i,2)}.
\end{aligned}
\tag{3.45}
$$

The $\boldsymbol{y}^1, \boldsymbol{y}^2$ and $\boldsymbol{y}^r$ determine the edge-capacity vector $\boldsymbol{z}$, and they together with $\boldsymbol{N}$ determine the vertex-demand vectors $\boldsymbol{d}^1$ and $\boldsymbol{d}^2$.

Since we rearranged the rows and columns of $\boldsymbol{H}$, we need do the same rearrangement for $\boldsymbol{M}$. After rearranging row and columns of $\boldsymbol{M}$, $\boldsymbol{M}$ is of the form that, column $2i-1$ and column $2i$ correspond to the $\boldsymbol{y}^1$-flow and $\boldsymbol{y}^2$-flow of the ith edge, respectively, and row $2i-1$ and row $2i$ correspond to the $\boldsymbol{u}$-coordinate and $\boldsymbol{v}$-coordinate of vertex i, respectively. In $\boldsymbol{M}$, each edge has a 2×2 identity matrix for one endpoint, and a negative 2×2 identity matrix for the other endpoint.

By the above setting, we can check that

$$
\boldsymbol{M}\boldsymbol{H}^{-1}\boldsymbol{M}^\top = \widehat{\boldsymbol{B}}^\top \boldsymbol{W} \widehat{\boldsymbol{B}}. \tag{3.46}
$$

Note that $\boldsymbol{M}$ has dimension $2n \times 2m$, and $\widehat{\boldsymbol{B}}$ has dimension $3m \times 2n$.

Secondly, we make the right hand side vectors of the two linear systems in Equa-

tion (3.40) and (3.41) to be the same, that is,

$$-t\boldsymbol{M}\boldsymbol{H}^{-1}\boldsymbol{g} = \widehat{\boldsymbol{B}}^{\top}\boldsymbol{W}^{1/2}\boldsymbol{c}^{\mathrm{B}^{>0}}, \tag{3.47}$$

by choosing the cost vector $\boldsymbol{c}$ properly. The gradient $\boldsymbol{g}$ is given by:

$$\boldsymbol{g} = \boldsymbol{c} - \frac{1}{t}\sum_{i\in[m]} \frac{1}{\boldsymbol{y}_i^1}\boldsymbol{e}_{2i-1} + \frac{1}{\boldsymbol{y}_i^2}\boldsymbol{e}_{2i} - \frac{1}{\boldsymbol{y}_i^r}\left(\boldsymbol{e}_{2i-1} + \boldsymbol{e}_{2i}\right).$$

For simplicity, let

$$\boldsymbol{f}(\boldsymbol{y}) \stackrel{\text{def}}{=} t\left(\boldsymbol{c} - \boldsymbol{g}\right).$$

$\boldsymbol{f}(\boldsymbol{y})$ is a vector in $2m$-dimension such that, the ith edge corresponds to the 2×1 sub-vector

$$\begin{pmatrix} \frac{1}{\boldsymbol{y}_i^1} - \frac{1}{\boldsymbol{y}_i^r} \\ \frac{1}{\boldsymbol{y}_i^2} - \frac{1}{\boldsymbol{y}_i^r} \end{pmatrix}.$$

Then we have

$$-t\boldsymbol{M}\boldsymbol{H}^{-1}\boldsymbol{g} = \boldsymbol{M}\boldsymbol{H}^{-1}\left(\boldsymbol{f} - t\boldsymbol{c}\right).$$

Recall the right hand side vector of Equation (3.47) is $\widehat{\boldsymbol{B}}^{\top}\boldsymbol{W}^{1/2}\boldsymbol{c}^{\mathrm{B}^{>0}}$. According to our construction in REDUCE $\mathcal{G}_{\mathrm{z},2}TO\mathcal{MC}_2$ (Algorithm 1) and REDUCE$\mathcal{MC}_2$TO$\mathcal{MC}_2^{>0}$ (Algorithm 4), $\boldsymbol{c}^{\mathrm{B}^{>0}} = (\boldsymbol{c}^{\mathrm{A}}; \boldsymbol{0})$, where $\boldsymbol{c}^{\mathrm{A}}$ is the right hand side vector of the general linear system instance. Note all the nonzero entries of $\boldsymbol{c}^{\mathrm{B}^{>0}}$ correspond to the type 1 edges in the main constraint set $\mathcal{A}$, see line 16 of Algorithm 1.

Recall the structure of $\boldsymbol{M}$ in Equation (3.42), the edge-vertex incidence matrix $\boldsymbol{N}$ contains all the m edges of the underlying graphs, which are exactly the set of edges in $\boldsymbol{B}$ constructed in REDUCE $\mathcal{G}_{\mathrm{z},2}TO\mathcal{MC}_2$ (Algorithm 1). Note for each edge of $\boldsymbol{N}$, its large-weight edge can be any of the corresponding type 1, type 2 and type $1+2$ edges. By Claim 3.8.5, we can partition the underlying graph edges into $E_1 \cup E_2 \cup E_{1+2}$ such that E_k

contains all the underlying edges whose large-weight edges are the corresponding type k edges, $k \in \{1, 2, 1+2\}$. Note $\boldsymbol{c}^{\mathrm{B}^{>0}}_{(i,1)} \neq 0$ only if edge i is in E_1.

We define a vector $\boldsymbol{c}' \in \mathbb{R}^{2m}$ such that the entry of $\boldsymbol{c}'$ which corresponds to the $\boldsymbol{y}^1$-flow of the ith edge equals to $w^{1/2}_{(i,1)} \boldsymbol{c}^{\mathrm{B}^{>0}}_{(i,1)}$. This definition gives:

$$\widehat{\boldsymbol{B}}^{\top} \boldsymbol{W}^{1/2} \boldsymbol{c}^{\mathrm{B}^{>0}} = \boldsymbol{M}\, \boldsymbol{c}'. \tag{3.48}$$

By Equation (3.47),

$$\boldsymbol{M}\, \boldsymbol{H}^{-1} (\boldsymbol{f} - t\boldsymbol{c}) = \boldsymbol{M}\, \boldsymbol{c}'.$$

Rearranging it,

$$\boldsymbol{M}\, \boldsymbol{H}^{-1} (\boldsymbol{f} - t\boldsymbol{c} - \boldsymbol{H}\, \boldsymbol{c}') = \boldsymbol{0}.$$

According to $\mathcal{MC}_2$GADGET (Algorithm 2), each gadget has a cycle containing both the 2 edges in E_{1+2}: $(t+3, t+4, t+2, t+5, t+6, t+1, t+3)$. Let E' be the set of all the edges in such gadget cycles. Let $\boldsymbol{p} \in \{0,1\}^m$ such that each entry corresponding to an edge in E' has value 1. Let $\boldsymbol{q} \stackrel{\text{def}}{=} (\boldsymbol{p}; \boldsymbol{p})$. According to REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{MC}_2$ (Algorithm 1), every gadget edge i has $\boldsymbol{c}'_i = 0$. Thus, we are free to choose the sign of a column of $\boldsymbol{N}^{\top}$ which corresponds to a gadget edge, which does not change Equations (3.46) and (3.48). [3] Without loss of generality, we assume that after proper sign flipping, $\boldsymbol{M}\, \boldsymbol{q} = \boldsymbol{0}$. It gives that

$$\boldsymbol{M}\, \boldsymbol{H}^{-1} (\boldsymbol{f} - t\boldsymbol{c} - \boldsymbol{H}\, \boldsymbol{c}' + \boldsymbol{H}\, \boldsymbol{q}) = \boldsymbol{0}.$$

We set $t\boldsymbol{c} = \boldsymbol{f} - \boldsymbol{H}\, \boldsymbol{c}' + \lambda \boldsymbol{H}\, \boldsymbol{q}$, where $\lambda > 0$ is a sufficiently large constant to be determined later.

For each edge i, let $\mathcal{I}_i$ be an indicator whose value is 1 if edge i in the set E' and 0

[3] Note the demand vectors $\boldsymbol{d}^1$ and $\boldsymbol{d}^2$ change after we change $\boldsymbol{N}$.

otherwise. For the ith edge, its corresponding 2×1 sub-vector in $t\boldsymbol{c}$ is

$$t \begin{pmatrix} \boldsymbol{c}_i \\ \boldsymbol{c}_{m+i} \end{pmatrix} = \begin{pmatrix} \frac{1}{\boldsymbol{y}_i^1} - \frac{1}{\boldsymbol{y}_i^r} \\ \frac{1}{\boldsymbol{y}_i^2} - \frac{1}{\boldsymbol{y}_i^r} \end{pmatrix} - \boldsymbol{c}'_i \begin{pmatrix} \frac{1}{(\boldsymbol{y}_i^1)^2} + \frac{1}{(\boldsymbol{y}_i^r)^2} \\ \frac{1}{(\boldsymbol{y}_i^r)^2} \end{pmatrix} + \mathcal{I}_i \lambda \boldsymbol{s}^i,$$

where

$$\boldsymbol{s}^i \stackrel{\text{def}}{=} \begin{pmatrix} \frac{1}{(\boldsymbol{y}_i^1)^2} + \frac{2}{(\boldsymbol{y}_i^r)^2} \\ \frac{1}{(\boldsymbol{y}_i^2)^2} + \frac{2}{(\boldsymbol{y}_i^r)^2} \end{pmatrix}.$$

Note that $\boldsymbol{s}^i \geq \boldsymbol{0}$ (entry-wise). Plugging the solution of $\boldsymbol{y}$ from Equation (3.45) gives:

$$t \begin{pmatrix} \boldsymbol{c}_i \\ \boldsymbol{c}_{m+i} \end{pmatrix} = \begin{pmatrix} \sqrt{\frac{w_{(i,2)}}{\gamma}} - \sqrt{\frac{w_{(i,1+2)}}{\gamma}} \\ \sqrt{\frac{w_{(i,1)}}{\gamma}} - \sqrt{\frac{w_{(i,1+2)}}{\gamma}} \end{pmatrix} - \boldsymbol{c}'_i \begin{pmatrix} \frac{w_{(i,2)}+w_{(i,1+2)}}{\gamma} \\ \frac{w_{(i,1+2)}}{\gamma} \end{pmatrix} + \mathcal{I}_i \lambda \boldsymbol{s}^i,$$

where $\gamma \stackrel{\text{def}}{=} w_{(i,1)}w_{(i,1+2)} + w_{(i,1)}w_{(i,2)} + w_{(i,2)}w_{(i,1+2)}$.

By Claim 3.8.5, each edge of the underlying graph has exactly one type of edges with large weight. We deal with each of these cases separately under the condition that two of the edge's weights are much smaller than that of the third one. We will also denote this ratio using $\epsilon > 0$:

1. If $w_{(i,2)} = w_{(i,1+2)} = \epsilon w_{(i,1)}$, then

$$t \begin{pmatrix} \boldsymbol{c}_i \\ \boldsymbol{c}_{m+i} \end{pmatrix} \geq -\boldsymbol{c}'_i \begin{pmatrix} \frac{2\epsilon}{\mu} \\ \frac{\epsilon}{\mu} \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1-\sqrt{\epsilon}}{\sqrt{\mu}} \end{pmatrix},$$

where $\mu = (2\epsilon + \epsilon^2)w_{(i,1)}$. This corresponds to an edge of E_1. If $\boldsymbol{c}'_i \leq 0$, then $t\boldsymbol{c}_i, t\boldsymbol{c}_{m+i} \geq 0$. If $\boldsymbol{c}'_i > 0$, then $\boldsymbol{c}'_i = w_{(i,1)}^{1/2} \boldsymbol{c}_{(i,1)}^{\mathrm{B}>0}$ where $\boldsymbol{c}_{(i,1)}^{\mathrm{B}>0}$ comes from an entry of $\boldsymbol{c}^{\mathrm{A}}$. We multiply -1 on both sides of the corresponding equation in the general linear system instance $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}^{\mathrm{A}}$ which creates this edge i, and will get $\boldsymbol{c}'_i < 0$ instead. [4]

[4]We first flip signs of equations of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}^{\mathrm{A}}$ so that $\boldsymbol{c}^{\mathrm{A}} \leq \boldsymbol{0}$ (entry-wise), which determines $\boldsymbol{N}$, $\boldsymbol{M}$ and $\boldsymbol{c}'$. We then flip signs of columns of $\boldsymbol{N}$, which we guarantee does not violate Equation (3.46) and (3.48).

2. If $w_{(i,1)} = w_{(i,1+2)} = \epsilon w_{(i,2)}$, then

$$t \begin{pmatrix} \boldsymbol{c}_i \\ \boldsymbol{c}_{m+i} \end{pmatrix} \geq -\boldsymbol{c}'_i \begin{pmatrix} \frac{1+\epsilon}{\mu} \\ \frac{\epsilon}{\mu} \end{pmatrix} + \begin{pmatrix} \frac{1-\sqrt{\epsilon}}{\sqrt{\mu}} \\ 0 \end{pmatrix},$$

where $\mu = (2\epsilon + \epsilon^2) w_{(i,2)}$. This corresponds to an edge of E_2, in which $\boldsymbol{c}'_i = 0$. We can check that $t\boldsymbol{c}_i, t\boldsymbol{c}_{m+i} \geq 0$.

3. If $w_{(i,1)} = w_{(i,2)} = \epsilon w_{(i,1+2)}$, then

$$t \begin{pmatrix} \boldsymbol{c}_i \\ \boldsymbol{c}_{m+i} \end{pmatrix} = -\boldsymbol{c}'_i \begin{pmatrix} \frac{1+\epsilon}{\mu} \\ \frac{1}{\mu} \end{pmatrix} + \begin{pmatrix} \frac{\sqrt{\epsilon}-1}{\sqrt{\mu}} \\ \frac{\sqrt{\epsilon}-1}{\sqrt{\mu}} \end{pmatrix} + \mathcal{I}_i \lambda \boldsymbol{s}^i,$$

where $\mu = (2\epsilon + \epsilon^2) w_{(i,1+2)}$. This corresponds to an edge of E_{1+2}, in which $\boldsymbol{c}'_i = 0$ and $\mathcal{I}_i = 1$. We choose λ such that, for every edge i in this case, $t\boldsymbol{c}_i, t\boldsymbol{c}_{m+i} \geq 0$.

□

Note in IPMs, the values of t is a sequence of increasing values. Whenever $t \geq m/\epsilon'$, the optimality gap is smaller than ϵ'. This ensures that the cost vector $\boldsymbol{c}$'s entries are in a reasonable range.

3.8.3 Isotropic Total Variation Minimization

For the isotropic total variation problem, we follow the formulations given in [32], namely given a graph $G = (V, E, \boldsymbol{w})$, we partition the sets into $S_1 \ldots S_k$, and minimize the objective

$$\|\boldsymbol{y} - \boldsymbol{s}\|_2^2 + \sum_{1 \leq i \leq k} \boldsymbol{w}_i \sqrt{\sum_{(u,v) \in S_i} \boldsymbol{w}_{uv} (\boldsymbol{y}_u - \boldsymbol{y}_v)^2}.$$

where $\boldsymbol{s}$ is the input signal vector. Let $\boldsymbol{N}$ be the edge-vertex incidence matrix of the graph G, then the dual of the grouped flow problem is:

$$\begin{aligned} \max \quad & \boldsymbol{s}^\top \boldsymbol{N}^\top \boldsymbol{f} \\ \text{subject to:} \quad & \sum_{e \in S_i} \boldsymbol{w}_e^{-1} \boldsymbol{f}_e^2 \leq \boldsymbol{w}_i^{-1} \qquad \forall 1 \leq i \leq k \end{aligned}$$

Both the primal and dual problems can be formulated into log barriers via second order cones [76]. In the case with minimizing vertex labels, we introduce a variable $\boldsymbol{y}_i$ for each cluster i, and instead minimize $\sum_i \boldsymbol{w}_i \boldsymbol{y}_i$ subject to the constraint

$$\boldsymbol{y}_i^2 \geq \sum_{(u,v) \in S_i} \boldsymbol{w}_e^{-1} (\boldsymbol{y}_u - \boldsymbol{y}_v)^2,$$

which in turn leads to the logarithmic barrier function

$$\log \left(\boldsymbol{y}_i^2 - \sum_{(u,v) \in S_i} \boldsymbol{w}_e^{-1} (\boldsymbol{y}_u - \boldsymbol{y}_v)^2 \right),$$

while in the flow case the barrier functions for each set of edges S_i is

$$\log \left(\boldsymbol{w}_i^{-1} - \sum_{e \in S_i} \boldsymbol{w}_e^{-1} \boldsymbol{f}_e^2 \right).$$

Due to the connections with the multicommodity flow problems, we will only state the connections through the flow version here. Recall that linear systems in $\mathcal{MC}_2$ have the form:

$$\boldsymbol{L}^1 \otimes \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \boldsymbol{L}^2 \otimes \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} + \boldsymbol{L}^{1+2} \otimes \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},$$

where E_1, E_2, E_{1+2} denotes the edges in $\boldsymbol{L}^1, \boldsymbol{L}^2, \boldsymbol{L}^{1+2}$, respectively. Edges in $E_1 \cup E_2$

correspond to individual edges, so it remains to show that edges in E_{1+2} can correspond to the Hessian matrices of clusters containing pairs of edges. In a cluster with two edges $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$, the gradient of the function

$$\log\left(\alpha - \boldsymbol{w}_1^{-1}\boldsymbol{f}_1^2 - \boldsymbol{w}_2^{-1}\boldsymbol{f}_2^2\right)$$

is

$$\frac{2}{r}\begin{pmatrix} -\boldsymbol{w}_1^{-1}\boldsymbol{f}_1 \\ -\boldsymbol{w}_2^{-1}\boldsymbol{f}_2 \end{pmatrix},$$

where we treat

$$r \stackrel{\text{def}}{=} \alpha - \boldsymbol{w}_1^{-1}\boldsymbol{f}_1^2 - \boldsymbol{w}_2^{-1}\boldsymbol{f}_2^2$$

as a new free variable because we are free to choose α in the IPM instance. Differentiating again (via the product rule) then gives the Hessian matrix:

$$\frac{-4}{r^2}\begin{pmatrix} \boldsymbol{w}_1^{-2}\boldsymbol{f}_1^2 & \boldsymbol{w}_1^{-1}\boldsymbol{w}_2^{-2}\boldsymbol{f}_1\boldsymbol{f}_2 \\ \boldsymbol{w}_1^{-1}\boldsymbol{w}_2^{-2}\boldsymbol{f}_1\boldsymbol{f}_2 & \boldsymbol{w}_2^{-2}\boldsymbol{f}_2^2 \end{pmatrix} + \frac{-2}{r}\begin{pmatrix} \boldsymbol{w}_1^{-1} & 0 \\ 0 & \boldsymbol{w}_2^{-1} \end{pmatrix}.$$

As we are free to choose $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, the second matrix can be any diagonal matrix. Then the freedom in choosing $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ then means the first matrix can be any rank 1 object. Therefore, this Hessian is equivalent to the block Hessian for two commodity flows generated in Equation (3.43) (up to a change of sign, since we are dealing with a maximization problem here), and its inverse as given in Equation (3.44) provides the characterization from Definition 2.3.4. We can also check more directly that the $1 + 2$ edge can be represented as one of these Hessian inverse blocks.

Claim 3.8.6. *For each edge (i, j) in E_{1+2}, there exist an edge-vertex incidence matrix $\boldsymbol{N}$,*

a diagonal matrix $\boldsymbol{W}$ *and a vector* $\boldsymbol{r}$ *such that* $\boldsymbol{W} \succcurlyeq \boldsymbol{r}\boldsymbol{r}^\top$ *and*

$$\boldsymbol{L}_{ij}^{1+2} \otimes \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} = \boldsymbol{N}^\top \left(\boldsymbol{W} - \boldsymbol{r}\boldsymbol{r}^\top \right) \boldsymbol{N}.$$

Proof. For simplicity, we remove all zero rows and columns of $\boldsymbol{L}_{ij}^{1+2}$ so that the 2-commodity matrix for edge (i, j) only has dimension 4×4. We pick

$$\boldsymbol{N} = \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{pmatrix}, \boldsymbol{W} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \text{ and } \boldsymbol{r} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

We check the PSD condition,

$$\boldsymbol{W} - \boldsymbol{r}\boldsymbol{r}^\top = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \succcurlyeq \boldsymbol{0}.$$

Then, we check that this decomposition equals to the 2-commodity matrix

$$\boldsymbol{N}^\top \left(\boldsymbol{W} - \boldsymbol{r}\boldsymbol{r}^\top \right) \boldsymbol{N} = \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & -1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix} = \boldsymbol{L}_{ij}^{1+2} \otimes \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

□

Proof of Lemma 3.2.9. Since the linear system related to the isotropic total variation minimization matrix is the same as the linear system for 2-commodity, all complexity parameters of these two linear systems are the same. □

CHAPTER 4

HARDNESS RESULTS OF PACKING LPS

In this chapter, we prove Theorem 1.1.2 and Theorem 1.1.3 by proving the following theorems.

Theorem 4.0.7. *If one can $(1-\epsilon)$-approximately solve a packing LP of dimension $n \times n$ in time $O(n^{\beta}/\epsilon^{\alpha})$ for some constant $\alpha, \beta > 0$, then one can solve any linear system instance $\text{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$ of dimension $O(n) \times O(n)$ in time $\widetilde{O}(n^{\beta+2\alpha}\kappa^{2\alpha}(\boldsymbol{A})\log(1/\epsilon))$, where $\kappa(\boldsymbol{A})$ is the condition number.*

Considering $\beta = 2$ and $\alpha = 0.0009$, Theorem 4.0.7 states that: $(1-\epsilon)$-approximately solving a packing LP of dimension $n \times n$ in time $O(n^2/\epsilon^{0.0009})$ would imply approximately solving a linear system of dimension $n \times n$ and condition number n^{10} in time $O(n^{2.1}\log(1/\epsilon))$. Given the best known bound on the matrix multiplication $\omega < 2.373$, this would prove Theorem 1.1.2.

Theorem 4.0.8. *If one can $(1-\epsilon)$-approximately solve a packing LP with N non-zeros in time $O(N/\epsilon^{\alpha})$ for some constant $\alpha > 0$, then one can solve any linear system instance $\text{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$ in time $\widetilde{O}(\text{nnz}(\boldsymbol{A})^{1+1.5\alpha}\kappa^{2\alpha}(\boldsymbol{A})\log(1/\epsilon))$, where $\kappa(\boldsymbol{A})$ is the condition number.*

Consider $\alpha = 0.22$, Theorem 4.0.8 states that: $(1-\epsilon)$-approximately solving a packing LP with N non-zeros in time $O(N/\epsilon^{0.22})$ would imply approximately solving a linear system with N non-zeros and condition number $\kappa = n^{1.5}$ in time $o(N\kappa\log(1/\epsilon))$. This is asymptotically faster than Conjugate Gradient, and thus proves Theorem 1.1.3. Note here we do not require $\boldsymbol{A}$ to be symmetric and PSD, Conjugate Gradient in fact solves $\boldsymbol{A}^{\top}\boldsymbol{A}$. Since $\sqrt{\kappa(\boldsymbol{A}^{\top}\boldsymbol{A})} = \kappa(\boldsymbol{A})$, the above running time is consistent to the usually used running time of CG.

We remark that for simplicity our proofs only consider sequential solvers. We believe that the same ideas work for parallel solvers.

4.1 Reducing a Linear System to a Packing LP

In this section we discuss how to reduce solving linear systems to solving packing LPs. We introduce two reductions from linear equation problems to packing LP problems. Using the two reductions, we will prove Theorems 4.0.7 and 4.0.8 later in this section.

Normalizing a Linear System. Given a linear system instance $\text{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, we denote $\boldsymbol{p} \stackrel{\text{def}}{=} \boldsymbol{A}^\top \boldsymbol{b}$. WLOG we assume $\boldsymbol{A}, \boldsymbol{b}$ are normalized so that

$$\|\boldsymbol{A}\|_{\max} \stackrel{\text{def}}{=} \max_{i,j} |\boldsymbol{A}_{i,j}| = 1, \qquad \|\boldsymbol{p}\|_2 = 1.$$

This normalization can be done in $\text{nnz}(\boldsymbol{A})$ time. Given any ϵ-approximate solution $\boldsymbol{x}$ to the normalized linear system, that is, $\|\boldsymbol{A}\boldsymbol{x} - \Pi_A \boldsymbol{b}\|_2 \leq \epsilon \|\Pi_A \boldsymbol{b}\|_2$ (refer to Section 2.2), we can scale the solution $\boldsymbol{x}$ to get an ϵ-approximate solution to the original linear system. Moreover, the condition number $\kappa(A)$ is not changed by the normalization. Furthermore, as $\|\boldsymbol{A}\|_{\max} = 1$, we must have $\sigma_{\max}(\boldsymbol{A}) \geq 1$, which gives the following claim.

Claim 4.1.1. $\sigma_{\min}^{-1}(\boldsymbol{A}) \leq \kappa(\boldsymbol{A})$.

Note that $\text{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$ does not guarantee that $\boldsymbol{b}$ is in the column space of $\boldsymbol{A}$, which means that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is infeasible. As suggested in Section 2.2, we rewrite $\text{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$ as the following:

$$\begin{aligned} \boldsymbol{A}^\top \boldsymbol{z} &= \boldsymbol{p} \\ \boldsymbol{A}\boldsymbol{x} &= \boldsymbol{z} \end{aligned} \tag{4.1}$$

Here, recall that $\boldsymbol{p} = \boldsymbol{A}^\top \boldsymbol{b}$, and $\boldsymbol{z}, \boldsymbol{x}$ are unknown vector variables. Note Equation (4.1)

always has a solution:

$$\boldsymbol{z} = (\boldsymbol{A}\boldsymbol{A}^\top)^\dagger \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{b},\ \boldsymbol{x} = (\boldsymbol{A}^\top \boldsymbol{A})^\dagger \boldsymbol{A}^\top \boldsymbol{b}.$$

As explained in Section 2.2, $\boldsymbol{x}$ is a minimizer for the regression problem $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$. In addition, the following claim upper bounds the ℓ_1 norm of a solution of Equation (4.1).

Claim 4.1.2. *There exists $\boldsymbol{x}^*, \boldsymbol{z}^*$ satisfying the linear system* (4.1)*, and*

$$\|\boldsymbol{z}^*\|_1 \leq \sqrt{m}\,\|\Pi_A \boldsymbol{b}\|_2\,, \qquad \|\boldsymbol{x}^*\|_1 \leq \sqrt{n}\sigma_{\min}^{-1}(\boldsymbol{A})\,\|\Pi_A \boldsymbol{b}\|_2\,,$$

moreover, we know

$$\|\Pi_A b\|_2 \in \left[\frac{\sigma_{\min}(\boldsymbol{A})}{\sigma_{\max}^2(\boldsymbol{A})}\,\|\boldsymbol{p}\|_2\,, \frac{\sigma_{\max}(\boldsymbol{A})}{\sigma_{\min}^2(\boldsymbol{A})}\,\|\boldsymbol{p}\|_2\right].$$

Proof. Consider the solution where $\boldsymbol{z}^* = \Pi_A \boldsymbol{b} = \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{p}$ and $\boldsymbol{x}^* = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$. It is straightforward to check $\boldsymbol{z}^*, \boldsymbol{x}^*$ satisfy (4.1). Moreover,

$$\|\boldsymbol{z}^*\|_1 \leq \sqrt{m}\,\|\boldsymbol{z}^*\|_2 = \sqrt{m}\,\|\Pi_A \boldsymbol{b}\|_2\,,$$

and

$$\|\boldsymbol{x}^*\|_1 \leq \sqrt{n}\,\|\boldsymbol{x}^*\|_2 \leq \sqrt{n}\sigma_{\min}^{-1}(\boldsymbol{A})\,\|\Pi_A \boldsymbol{b}\|_2$$

where the last inequality follows from $\boldsymbol{A}\boldsymbol{x}^* = \Pi_A \boldsymbol{b}$.

Finally, the upper and lower bounds of $\|\Pi_A \boldsymbol{b}\|_2$ follow from $\Pi_A \boldsymbol{b} = \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{p}$.

□

Main Ideas. Given Equation (4.1), we first apply standard tricks to turn each equality into two inequalities, and make all variables non-negative. For each variable $\boldsymbol{x}_j$ ($\boldsymbol{z}_i$), we

create two non-negative variables $\boldsymbol{x}_j^{(+)}, \boldsymbol{x}_j^{(-)}$ ($\boldsymbol{z}_i^{(+)}, \boldsymbol{z}_i^{(-)}$, respectively) such that:

$$\begin{aligned}\boldsymbol{x}_j &= \boldsymbol{x}_j^{(+)} - \boldsymbol{x}_j^{(-)} \\ \boldsymbol{z}_i &= \boldsymbol{z}_i^{(+)} - \boldsymbol{z}_i^{(-)}\end{aligned}$$

Let $\boldsymbol{x}^{(+)}$, $\boldsymbol{x}^{(-)}$, $\boldsymbol{z}^{(+)}$, and $\boldsymbol{z}^{(-)}$ be vectors whose entries are $\boldsymbol{x}_j^{(+)}$, $\boldsymbol{x}_j^{(-)}$, $\boldsymbol{z}_i^{(+)}$, $\boldsymbol{z}_i^{(-)}$ respectively. We define LP($\boldsymbol{A}, \boldsymbol{b}$) as the feasibility LP of finding a solution satisfying:

$$\begin{aligned}\boldsymbol{A}^T\boldsymbol{z}^{(+)} - \boldsymbol{A}^T\boldsymbol{z}^{(-)} &\leq \boldsymbol{p} \\ -\boldsymbol{A}^T\boldsymbol{z}^{(+)} + \boldsymbol{A}^T\boldsymbol{z}^{(-)} &\leq -\boldsymbol{p} \\ \boldsymbol{A}\boldsymbol{x}^{(+)} - \boldsymbol{A}\boldsymbol{x}^{(-)} - \left(\boldsymbol{z}^{(+)} - \boldsymbol{z}^{(-)}\right) &\leq 0 \\ -\left(\boldsymbol{A}\boldsymbol{x}^{(+)} - \boldsymbol{A}\boldsymbol{x}^{(-)}\right) + \left(\boldsymbol{z}^{(+)} - \boldsymbol{z}^{(-)}\right) &\leq 0 \\ \boldsymbol{x}^{(+)}, \boldsymbol{x}^{(-)}, \boldsymbol{z}^{(+)}, \boldsymbol{z}^{(-)} &\geq 0\end{aligned}$$

For simplicity, we use

$$\widetilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} \leq \tilde{\boldsymbol{b}},\ \tilde{\boldsymbol{x}} \geq 0 \tag{4.2}$$

to denote the above feasibility LP, and explain how to convert all entries of $\widetilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}$ to be non-negative. We will use a *bounding box* constraint: $\sum_{i=1}^{n} \tilde{\boldsymbol{x}}_i \leq U$, where U is an upper bound of the ℓ_1 norm of some feasible solution $\tilde{\boldsymbol{x}}$. We will show in Claim 4.1.2 that U is not too large. This is equivalent to the following equality with a new slack variable $x_0 \geq 0$:

$$\boldsymbol{x}_0 + \sum_{i=1}^{n} \tilde{\boldsymbol{x}}_i = U. \tag{4.3}$$

Adding this equality to each constraint of $\widetilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} \leq \tilde{\boldsymbol{b}}$ makes all entries of the LP non-negative, given that all entries of $\widetilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{b}}$ are between $[-1, 1]$. Moreover, this operation does not change the feasible solutions of LP (4.2) under the bounding box constraint.

However, we cannot have the equality (4.3) as a constraint, since it violates the form of packing LPs. Instead, we add the inequality $\boldsymbol{x}_0 + \sum_{i=1}^{n} \tilde{\boldsymbol{x}}_i \leq U$ as a constraint, and use

$$\max\ \boldsymbol{x}_0 + \sum_{i=1}^{n} \tilde{\boldsymbol{x}}_i$$

as the objective function. Note there always exists a feasible solution $\boldsymbol{x}_0, \tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n$ with objective value U, which forces $\boldsymbol{x}_0 + \sum_{i=1}^{n} \tilde{\boldsymbol{x}}_i = U$ holds for optimal solutions. Thus, an optimal solution of the above packing LP is a solution for $\widetilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}$.

We will show in Section 4.1.1, the above idea gives a reduction from an arbitrary linear system instance to a packing LP instance such that, the dimensions, and accuracy parameters of the two instances are equivalent up to polynomially factors. In Section 4.1.2, we will present a slightly modified reduction which preserves the sparsity of a linear system instance. Finally, in Section 4.1.3, we will use the standard technique *iterative refinement* for solving linear systems to reduce the polynomially error dependence into logarithmic dependence.

4.1.1 Dense Reduction $\text{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$

$\text{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$ takes inputs $\boldsymbol{A}$, $\boldsymbol{b}$ and a tight upper bound of $\|\boldsymbol{z}^*\|_1 + \|\boldsymbol{x}^*\|_1$, and outputs a packing LP.

To turn $\text{LP}(\boldsymbol{A}, \boldsymbol{b})$ into a packing LP, we add one non-negative slack variable $s \geq 0$, and denote

$$\alpha_{\text{sum}} \stackrel{\text{def}}{=} s + \sum_{1 \leq j \leq n} \left(\boldsymbol{x}_j^{(+)} + \boldsymbol{x}_j^{(-)} \right) + \sum_{1 \leq i \leq m} \left(\boldsymbol{z}_i^{(+)} + \boldsymbol{z}_i^{(-)} \right).$$

Note α_{sum} is merely a short-hand for the sum of all the variables instead of a new variable. For each existing constraint of $\text{LP}(\boldsymbol{A}, \boldsymbol{b})$ except the positivity constraints on single variables, we add α_{sum} and U to the LHS and RHS of the constraint respectively, and we also add

$$\alpha_{\text{sum}} \leq U \tag{4.4}$$

as an additional constraint. This completes the construction of $\text{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$'s constraints, and if written explicitly are the following

$$s + \sum_{1\le i\le m} (1 + \boldsymbol{A}_{ij})\, \boldsymbol{z}_i^{(+)} + (1 - \boldsymbol{A}_{ij})\, \boldsymbol{z}_i^{(-)} + \sum_{1\le j'\le n} \boldsymbol{x}_{j'}^{(+)} + \boldsymbol{x}_{j'}^{(-)} \le \boldsymbol{p}_j + U \qquad \forall_{1\le j\le n}$$

$$s + \sum_{1\le i\le m} (1 - \boldsymbol{A}_{ij})\, \boldsymbol{z}_i^{(+)} + (1 + \boldsymbol{A}_{ij})\, \boldsymbol{z}_i^{(-)} + \sum_{1\le j'\le n} \boldsymbol{x}_{j'}^{(+)} + \boldsymbol{x}_{j'}^{(-)} \le -\boldsymbol{p}_j + U \qquad \forall_{1\le j\le n}$$

$$s + \sum_{1\le j\le n} (1 + \boldsymbol{A}_{ij})\, \boldsymbol{x}_j^{(+)} + (1 - \boldsymbol{A}_{ij})\, \boldsymbol{x}_j^{(-)} + 2\boldsymbol{z}_i^{(-)} + \sum_{1\le i'\le n, i'\ne i} \boldsymbol{z}_{i'}^{(+)} + \boldsymbol{z}_{i'}^{(-)} \le U \qquad \forall_{1\le i\le m}$$

$$s + \sum_{1\le j\le n} (1 - \boldsymbol{A}_{ij})\, \boldsymbol{x}_j^{(+)} + (1 + \boldsymbol{A}_{ij})\, \boldsymbol{x}_j^{(-)} + 2\boldsymbol{z}_i^{(+)} + \sum_{1\le i'\le n, i'\ne i} \boldsymbol{z}_{i'}^{(+)} + \boldsymbol{z}_{i'}^{(-)} \le U \qquad \forall_{1\le i\le m}$$

$$s + \sum_{1\le j\le n} \left(\boldsymbol{x}_j^{(+)} + \boldsymbol{x}_j^{(-)}\right) + \sum_{1\le i\le m} \left(\boldsymbol{z}_i^{(+)} + \boldsymbol{z}_i^{(-)}\right) \le U$$

$$s, \boldsymbol{x}^{(+)}, \boldsymbol{x}^{(-)}, \boldsymbol{z}^{(+)}, \boldsymbol{z}^{(-)} \ge 0$$

The objective of $\text{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$ is to maximize α_{sum}, that is

$$\max\ s + \sum_{1\le j\le n} \left(\boldsymbol{x}_j^{(+)} + \boldsymbol{x}_j^{(-)}\right) + \sum_{1\le i\le m} \left(\boldsymbol{z}_i^{(+)} + \boldsymbol{z}_i^{(-)}\right).$$

Lemma 4.1.3. *Given $\boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{b} \in \mathbb{R}^m$, $\text{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$ constructs a packing LP of size $O(m+n) \times O(m+n)$. Furthermore, if $U \ge \|\boldsymbol{x}^*\|_1 + \|\boldsymbol{z}^*\|$ for some feasible solution $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ of LP$(\boldsymbol{A}, \boldsymbol{b})$, then $\text{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$ has optimal value U; Otherwise, $\text{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$ is infeasible.*

Proof. The size of $\text{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$ is obvious. For it to be a packing LP, we need to show

1. All variables are non-negative: This holds by construction.
2. All coefficients are non-negative: This is true since $\|\boldsymbol{A}\|_{\max} \le 1$, so by adding 1 to every coefficient, all coefficients become non-negative.
3. All constants on the RHS of the constraints are non-negative: This is true since $\|\boldsymbol{p}\|_2 = 1$, so $\|\boldsymbol{p}\|_\infty \le 1$, and we add $U \ge 1$ to the RHS of each constraint.

4. All coefficients in the objective function are non-negative: This is true since all coefficients are 1 in the objective.

If $U > \|\boldsymbol{x}^*\|_1 + \|\boldsymbol{z}^*\|_1$ for any $\boldsymbol{x}^*$ and $\boldsymbol{z}^*$ statisfying the linear system (4.1), consider the following solution to $\text{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$ where we set $\boldsymbol{x}^{(+)}$ and $\boldsymbol{x}^{(-)}$ based on the signs of $\boldsymbol{x}^*$

$$\left(\boldsymbol{x}_i^{(+)}, \boldsymbol{x}_i^{(-)}\right) := \begin{cases} (\boldsymbol{x}^*{}_i, 0) & \text{if } \boldsymbol{x}^*{}_i \geq 0, \\ (0, -\boldsymbol{x}^*{}_i) & \text{if } \boldsymbol{x}^*{}_i \leq 0, \end{cases}$$

and similarly for $\boldsymbol{z}^{(+)}$ and $\boldsymbol{z}^{(-)}$:

$$\left(\boldsymbol{z}_i^{(+)}, \boldsymbol{z}_i^{(-)}\right) := \begin{cases} (\boldsymbol{z}^*{}_i, 0) & \text{if } \boldsymbol{z}^*{}_i \geq 0, \\ (0, -\boldsymbol{z}^*{}_i) & \text{if } \boldsymbol{z}^*{}_i \leq 0, \end{cases}$$

For the added slack variable s, as $U > \|\boldsymbol{x}^*\|_1 + \|\boldsymbol{z}^*\|_1$, we can set

$$s := U - \|\boldsymbol{x}^*\|_1 + \|\boldsymbol{z}^*\|_1 = U - \left(\sum_{1 \leq j \leq n} \left(\boldsymbol{x}_j^{(+)} + \boldsymbol{x}_j^{(-)} \right) + \sum_{1 \leq i \leq m} \left(\boldsymbol{z}_i^{(+)} + \boldsymbol{z}_i^{(-)} \right) \right).$$

It is straightforward to see this solution is feasible, and gives objective value U, which must be the optimal due to the constraint (4.4). □

In the following lemma, we translate the error bound between the original linear system and $\text{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$.

Lemma 4.1.4. *Consider an instance* $\text{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon')$ *and any number* U *such that* $LP(\boldsymbol{A}, \boldsymbol{x})$ *has a feasible solution with* ℓ_1 *norm at most* U*. Let* $\epsilon > 0$ *be any value satisfying*

$$\epsilon \leq \frac{\epsilon'}{(\sqrt{m} + \sqrt{n})U}.$$

Suppose we can compute a feasible solution $\boldsymbol{x}^{(+)}, \boldsymbol{x}^{(-)}, \boldsymbol{z}^{(+)}, \boldsymbol{z}^{(-)}, \boldsymbol{s}^{(p)}, \boldsymbol{s}^{(0)}$ *of* $PLP_{dense}(\boldsymbol{A}, \boldsymbol{b}, U)$ *with objective value at least* $(1 - \epsilon)\, U$*. Then,* $\boldsymbol{x} = \boldsymbol{x}^{(+)} - \boldsymbol{x}^{(-)}$ *is a solution to* $\text{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon')$*,*

that is,

$$\|\boldsymbol{A}\boldsymbol{x} - \Pi_A \boldsymbol{b}\|_2 \leq \epsilon' \|\Pi_A \boldsymbol{b}\|_2 .$$

Proof. To solve $\text{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon')$, we first check whether $\boldsymbol{A}^\top \boldsymbol{b} = \boldsymbol{0}$, that is, whether $\boldsymbol{b}$ is orthogonal to the column space of $\boldsymbol{A}$. If there is true, then $\boldsymbol{x} = \boldsymbol{0}$ is a solution to $\text{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon')$ and we have done. In the rest of the proof, we assume that $\boldsymbol{A}^\top \boldsymbol{b} \neq \boldsymbol{0}$, equivalently, $\Pi_A \boldsymbol{b} \neq \boldsymbol{0}$. By Lemma 2.2.5,

$$\|\Pi_A \boldsymbol{b}\|_2 \geq \frac{1}{\sigma_{\max}(\boldsymbol{A})} \geq 1. \tag{4.5}$$

Consider any feasible solution, and let $\boldsymbol{x} = \boldsymbol{x}^{(+)} - \boldsymbol{x}^{(-)}$, $\boldsymbol{z} = \boldsymbol{z}^{(+)} - \boldsymbol{z}^{(-)}$. We want to bound the error in $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{z}$, $\boldsymbol{A}^T \boldsymbol{z} = \boldsymbol{p}$. Consider the i-th equality constraint in the linear system: $\boldsymbol{A}_i \boldsymbol{x} = \boldsymbol{z}_i$, we have a corresponding pair of constraints in $\text{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$

$$\alpha_{\text{sum}} - \left(\boldsymbol{z}_i^{(+)} - \boldsymbol{z}_i^{(-)}\right) + \sum_{1 \leq j \leq n} \boldsymbol{A}_{ij} \left(\boldsymbol{x}_j^{(+)} - \boldsymbol{x}_j^{(-)}\right) \leq U$$
$$\alpha_{\text{sum}} + \left(\boldsymbol{z}_i^{(+)} - \boldsymbol{z}_i^{(-)}\right) - \sum_{1 \leq j \leq n} \boldsymbol{A}_{ij} \left(\boldsymbol{x}_j^{(+)} - \boldsymbol{x}_j^{(-)}\right) \leq U$$

Since we have a feasible solution, we know from the above two constraints that

$$|\boldsymbol{A}_i \boldsymbol{x} - \boldsymbol{z}_i| \leq U - \alpha_{\text{sum}}.$$

The same argument holds for any linear constraint $(\boldsymbol{A}^T)_j \boldsymbol{z} = \boldsymbol{p}_j$, so we have

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{z}\|_\infty , \left\|\boldsymbol{A}^T \boldsymbol{z} - \boldsymbol{p}\right\|_\infty \leq U - \alpha_{\text{sum}},$$

and by the approximation guarantee the RHS is at most $\epsilon \cdot U$.

Now we bound the error of the solution $\boldsymbol{x}$ in terms of the linear system

$$\begin{aligned}\|\boldsymbol{A}\boldsymbol{x} - \Pi_A \boldsymbol{b}\|_2 &\leq \|\boldsymbol{A}\boldsymbol{x} - \Pi_A \boldsymbol{z}\|_2 + \|\Pi_A \boldsymbol{z} - \Pi_A \boldsymbol{b}\|_2 \\ &\leq \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{z}\|_2 + \sigma_{\min}^{-1}(A) \left\|\boldsymbol{A}^T \boldsymbol{z} - \boldsymbol{A}^T \boldsymbol{b}\right\|_2 \\ &\leq \sqrt{m} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{z}\|_\infty + \sqrt{n}\sigma_{\min}^{-1}(A) \left\|\boldsymbol{A}^T \boldsymbol{z} - \boldsymbol{A}^T \boldsymbol{b}\right\|_\infty \\ &\leq \epsilon \sigma_{\min}^{-1}(A)(\sqrt{m} + \sqrt{n})U. \end{aligned} \tag{4.6}$$

In the second inequality, we have $\|\boldsymbol{A}\boldsymbol{x} - \Pi_A \boldsymbol{z}\|_2 \leq \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{z}\|_2$ because $\boldsymbol{A}\boldsymbol{x}$ is in the span of columns of $\boldsymbol{A}$, so projecting $\boldsymbol{z}$ to the column span of $\boldsymbol{A}$ makes the distance to $\boldsymbol{A}\boldsymbol{x}$ smaller. Also in the second inequality, we use

$$\begin{aligned}\|\Pi_A \boldsymbol{z} - \Pi_A \boldsymbol{b}\|_2 &= \left\|\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T(\boldsymbol{z} - \boldsymbol{b})\right\| \\ &= \sqrt{(\boldsymbol{z} - \boldsymbol{b})^T \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T(\boldsymbol{z} - \boldsymbol{b})} \\ &= \sqrt{(\boldsymbol{z} - \boldsymbol{b})^T \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T(\boldsymbol{z} - \boldsymbol{b})} \\ &\leq \sigma_{\min}^{-1}(A) \left\|\boldsymbol{A}^T \boldsymbol{z} - \boldsymbol{A}^T \boldsymbol{b}\right\|_2 .\end{aligned}$$

Plugging Equation (4.5) into Equation (4.6):

$$\|\boldsymbol{A}\boldsymbol{x} - \Pi_A \boldsymbol{b}\|_2 \leq \epsilon \sigma_{\min}(\boldsymbol{A})^{-1}(\sqrt{m} + \sqrt{n})U \cdot \|\Pi_A \boldsymbol{b}\|_2 .$$

Setting

$$\epsilon \leq \frac{\epsilon' \sigma_{\min}(\boldsymbol{A})}{(\sqrt{m} + \sqrt{n})U} \leq \frac{\epsilon'}{(\sqrt{m} + \sqrt{n})U},$$

we have $\|\boldsymbol{A}\boldsymbol{x} - \Pi_A \boldsymbol{b}\|_2 \leq \epsilon' \|\Pi_A \boldsymbol{b}\|$. □

By Claim 4.1.2 and Claim 4.1.1, with a $\log(\kappa(\boldsymbol{A}))$ overhead in running time, we can assume we have a almost tight bound U, for example $U \leq 2\sqrt{m+n}\sigma^{-2}(\boldsymbol{A})$. With this U,

by Lemma 4.1.4 and Claim 4.1.1, it suffices to set

$$\epsilon = \frac{\epsilon'}{100(m+n)\kappa^2(\boldsymbol{A})}.$$

4.1.2 Sparse Reduction $\text{PLP}_{\text{sparse}}(\boldsymbol{A}, \boldsymbol{b}, U)$

The sparsity preserving reduction follows the same approach as the dense reduction. However, instead of adding the same α_{sum} and U to the LHS and RHS of every inequality of $\text{LP}(\boldsymbol{A}, \boldsymbol{b})$ to make the coefficients non-negative, we will pin-point only the non-zero coefficients to preserve sparsity. We construct $\text{PLP}_{\text{sparse}}(\boldsymbol{A}, \boldsymbol{b}, U)$ starting from $\text{LP}(\boldsymbol{A}, \boldsymbol{b})$ as follows.

For each $j \in [1, n]$, look at the pair of constraints in $\text{LP}(\boldsymbol{A}, \boldsymbol{b})$ corresponding to the j-th row of $\boldsymbol{A}^T\boldsymbol{z} = \boldsymbol{p}$ in the linear system (4.1):

$$(\boldsymbol{A}^T)_j \boldsymbol{z}^{(+)} - (\boldsymbol{A}^T)_j \boldsymbol{z}^{(-)} \leq \boldsymbol{p}_j$$
$$-(\boldsymbol{A}^T)_j \boldsymbol{z}^{(+)} + (\boldsymbol{A}^T)_j \boldsymbol{z}^{(-)} \leq -\boldsymbol{p}_j$$

We add a non-negative slack variable $\boldsymbol{s}_j^{(p)}$ that serves as the slack for *both* of these constraints. Furthermore, we denote

$$\alpha_j^{(p)} := \boldsymbol{s}_j^{(p)} + \sum_{i:A_{ij}\neq 0} \boldsymbol{z}_i^{(+)} + \boldsymbol{z}_i^{(-)},$$

and again note $\alpha_j^{(p)}$ is merely a shorthand rather than a new variable. We add $\alpha_j^{(p)}$ and U to the LHS and RHS respectively of both the constraints above. Since $\|\boldsymbol{A}\|_{\max} \leq 1$, all the coefficients in these constraints become non-negative. We then add to $\text{PLP}_{\text{sparse}}(\boldsymbol{A}, \boldsymbol{b}, U)$ a new inequality

$$\alpha_j^{(p)} \leq U.$$

Similarly, for each $i \in [1, m]$, we consider the pair of constraints corresponding to the i-th

row in $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{z}$

$$\boldsymbol{A}_i\boldsymbol{x}^{(+)} - \boldsymbol{A}_i\boldsymbol{x}^{(-)} - (\boldsymbol{z}_i^{(+)} - \boldsymbol{z}_i^{(-)}) \leq 0$$
$$-\left(\boldsymbol{A}_i\boldsymbol{x}^{(+)} - \boldsymbol{A}_i\boldsymbol{x}^{(-)}\right) + \left(\boldsymbol{z}_i^{(+)} - \boldsymbol{z}_i^{(-)}\right) \leq 0$$

We add a non-negative slack variable $\boldsymbol{s}_i^{(0)}$ for both of these constraints, and denote

$$\alpha_i^{(0)} := \boldsymbol{s}_i^{(0)} + \boldsymbol{z}_i^{(+)} + \boldsymbol{z}_i^{(-)} + \sum_{1\leq j\leq n:\boldsymbol{A}_{ij}\neq 0} \boldsymbol{x}_j^{(+)} + \boldsymbol{x}_j^{(-)}.$$

We add $\alpha_i^{(0)}$ and U to the LHS and RHS of the pair of constraints, and add $\alpha_i^{(0)} \leq U$ as a new constraint.

We can write the constraints of $\text{PLP}_{\text{sparse}}(\boldsymbol{A}, \boldsymbol{b}, U)$ explicitly as follows.

$$\boldsymbol{s}_j^{(p)} + \sum_{1\leq i\leq m:\boldsymbol{A}_{ij}\neq 0} (1+\boldsymbol{A}_{ij})\,\boldsymbol{z}_i^{(+)} + (1-\boldsymbol{A}_{ij})\,\boldsymbol{z}_i^{(-)} \leq \boldsymbol{p}_j + U \qquad \forall_{1\leq j\leq n}$$
$$\boldsymbol{s}_j^{(p)} + \sum_{1\leq i\leq m:\boldsymbol{A}_{ij}\neq 0} (1-\boldsymbol{A}_{ij})\,\boldsymbol{z}_i^{(+)} + (1+\boldsymbol{A}_{ij})\,\boldsymbol{z}_i^{(-)} \leq -\boldsymbol{p}_j + U \qquad \forall_{1\leq j\leq n}$$
$$\boldsymbol{s}_j^{(p)} + \sum_{1\leq i\leq m:\boldsymbol{A}_{ij}\neq 0} \boldsymbol{z}_i^{(+)} + \boldsymbol{z}_i^{(-)} \leq U \qquad \forall_{1\leq j\leq n}$$
$$2\boldsymbol{z}_i^{(-)} + \boldsymbol{s}_i^{(0)} + \sum_{1\leq j\leq n:\boldsymbol{A}_{ij}\neq 0} (1+\boldsymbol{A}_{ij})\,\boldsymbol{x}_j^{(+)} + (1-\boldsymbol{A}_{ij})\,\boldsymbol{x}_j^{(-)} \leq U \qquad \forall_{1\leq i\leq m}$$
$$2\boldsymbol{z}_i^{(+)} + \boldsymbol{s}_i^{(0)} + \sum_{1\leq j\leq n:\boldsymbol{A}_{ij}\neq 0} (1-\boldsymbol{A}_{ij})\,\boldsymbol{x}_j^{(+)} + (1+\boldsymbol{A}_{ij})\,\boldsymbol{x}_j^{(-)} \leq U \qquad \forall_{1\leq i\leq m}$$
$$\boldsymbol{z}_i^{(+)} + \boldsymbol{z}_i^{(-)} + \boldsymbol{s}_i^{(0)} + \sum_{1\leq j\leq n:\boldsymbol{A}_{ij}\neq 0} \boldsymbol{x}_j^{(+)} + \boldsymbol{x}_j^{(-)} \leq U \qquad \forall_{1\leq i\leq m}$$
$$\boldsymbol{x}^{(+)}, \boldsymbol{x}^{(-)}, \boldsymbol{z}^{(+)}, \boldsymbol{z}^{(-)}, \boldsymbol{s}^{(p)}, \boldsymbol{s}^{(0)} \geq 0$$

Finally, the objective of $\text{PLP}_{\text{sparse}}(\boldsymbol{A}, \boldsymbol{b}, U)$ is to maximize $\sum_{1\leq j\leq n} \alpha_j^{(p)} + \sum_{1\leq i\leq m} \alpha_i^{(0)}$.

Lemma 4.1.5. *Given $\boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{b} \in \mathbb{R}^m$, $\text{PLP}_{\text{sparse}}(\boldsymbol{A}, \boldsymbol{b}, U)$ constructs a packing LP with $O(\text{nnz}(\boldsymbol{A}))$ non-zeros . Furthermore, if $U \geq \|\boldsymbol{x}^*\|_1 + \|\boldsymbol{z}^*\|$ for some feasible solution*

$(\boldsymbol{x}^*, \boldsymbol{z}^*)$ *of LP*$(\boldsymbol{A}, \boldsymbol{b})$*, then PLP*$_{sparse}(\boldsymbol{A}, \boldsymbol{b}, U)$ *has optimal value* $(m+n)U$*; Otherwise, PLP*$_{sparse}(\boldsymbol{A}, \boldsymbol{b}, U)$ *is infeasible.*

The proof is a straightforward but tedious adaptation of the proof of Lemma 4.1.3, so we omit it. Now we translate the error bound between the original linear system and $\text{PLP}_{\text{sparse}}(\boldsymbol{A}, \boldsymbol{b}, U)$.

Lemma 4.1.6. *Consider an instance* $\text{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon')$ *and any number* U *such that LP*$(\boldsymbol{A}, \boldsymbol{x})$ *has a feasible solution with* ℓ_1 *norm at most* U*. Let* $\epsilon > 0$ *be any value satisfying*

$$\epsilon \leq \frac{\epsilon'}{(m+n)U}.$$

Suppose we can compute a feasible solution $\boldsymbol{x}^{(+)}, \boldsymbol{x}^{(-)}, \boldsymbol{z}^{(+)}, \boldsymbol{z}^{(-)}, \boldsymbol{s}^{(p)}, \boldsymbol{s}^{(0)}$ *of PLP*$_{sparse}(\boldsymbol{A}, \boldsymbol{b}, U)$ *with objective value at least* $(1-\epsilon) \cdot (m+n)\, U$*. Then,* $\boldsymbol{x} = \boldsymbol{x}^{(+)} - \boldsymbol{x}^{(-)}$ *is a solution to* $\text{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon')$*, that is,*

$$\|\boldsymbol{A}\boldsymbol{x} - \Pi_A \boldsymbol{b}\|_2 \leq \epsilon' \|\Pi_A \boldsymbol{b}\|_2 .$$

Proof. By the same argument as the proof of Lemma 4.1.4, we assume that $\Pi_A \boldsymbol{b} \neq \boldsymbol{0}$ and thus $\|\Pi_A \boldsymbol{b}\|_2 \geq \frac{1}{\sigma_{\max}(\boldsymbol{A})} \geq 1$.

Given the $(1-\epsilon)$-approx optimal solution to $\text{PLP}_{\text{sparse}}(\boldsymbol{A}, \boldsymbol{b}, U)$, we consider the following solution to the linear system

$$\boldsymbol{x} := \boldsymbol{x}^{(+)} - \boldsymbol{x}^{(-)}$$
$$\boldsymbol{z} := \boldsymbol{z}^{(+)} - \boldsymbol{z}^{(-)}$$

We want to bound the error in $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{z}, \boldsymbol{A}^T\boldsymbol{z} = \boldsymbol{p}$.

Consider the i-th equality constraint in the linear system $\boldsymbol{A}_i\boldsymbol{x} = \boldsymbol{z}_i$, we have a corre-

sponding pair of constraints in $\text{PLP}_{\text{sparse}}(\boldsymbol{A}, \boldsymbol{b}, U)$:

$$\alpha_i^{(0)} + \boldsymbol{A}_i \boldsymbol{x}^{(+)} - \boldsymbol{A}_i \boldsymbol{x}^{(-)} - (\boldsymbol{z}_i^{(+)} - \boldsymbol{z}_i^{(-)}) \leq U$$
$$\alpha_i^{(0)} - \left(\boldsymbol{A}_i \boldsymbol{x}^{(+)} - \boldsymbol{A}_i \boldsymbol{x}^{(-)}\right) + \left(\boldsymbol{z}_i^{(+)} - \boldsymbol{z}_i^{(-)}\right) \leq U$$

which subtracting away $\alpha_i^{(0)}$ and U from the LHS and RHS respectively gives

$$|(\boldsymbol{A}\boldsymbol{x})_i - \boldsymbol{z}_i| \leq U - \alpha_i^{(0)}.$$

The same argument holds for any pair of linear constraints corresponding to $(\boldsymbol{A}^T \boldsymbol{z})_j = \boldsymbol{p}_j$, and in total we have

$$\begin{aligned}
&\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{z}\|_1 + \left\|\boldsymbol{A}^T \boldsymbol{z} - \boldsymbol{p}\right\|_1 \\
&\leq (m+n)\, U - \sum_j \alpha_j^{(p)} - \sum_i \alpha_i^{(0)} \\
&\leq \epsilon\, (m+n)\, U.
\end{aligned}$$

Now we bound the error of the solution $\boldsymbol{x}$ in terms of LSA

$$\begin{aligned}
\|\boldsymbol{A}\boldsymbol{x} - \Pi_A \boldsymbol{b}\|_2 &\leq \|\boldsymbol{A}\boldsymbol{x} - \Pi_A \boldsymbol{z}\|_2 + \|\Pi_A \boldsymbol{z} - \Pi_A \boldsymbol{b}\|_2 \\
&\leq \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{z}\|_2 + \sigma_{\min}^{-1}(A) \left\|\boldsymbol{A}^T \boldsymbol{z} - \boldsymbol{A}^T \boldsymbol{b}\right\|_2 \\
&\leq \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{z}\|_1 + \sigma_{\min}^{-1}(A) \left\|\boldsymbol{A}^T \boldsymbol{z} - \boldsymbol{A}^T \boldsymbol{b}\right\|_1 \\
&\leq \epsilon \sigma_{\min}^{-1}(A)(m+n)U \\
&\leq \epsilon \sigma_{\min}^{-1}(A)(m+n)U \cdot \|\Pi_A \boldsymbol{b}\|_2 .
\end{aligned}$$

Setting

$$\epsilon \leq \frac{\epsilon'}{(m+n)U},$$

we have $\|\boldsymbol{A}\boldsymbol{x} - \Pi_A \boldsymbol{b}\|_2 \leq \epsilon' \|\Pi_A \boldsymbol{b}\|_2$. □

Similar to the dense reduction, by a standard binary search on U, it suffices to set

$$\epsilon = \frac{\epsilon'}{100(m^{3/2} + n^{3/2})\kappa^2(\boldsymbol{A})}.$$

4.1.3 Reducing $\mathrm{poly}(1/\epsilon)$-Dependence to $\log(1/\epsilon)$-Dependence

As in Lemma 4.1.4 and Lemma 4.1.6, to solve a linear system instance $\mathrm{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon')$, it suffices to solve a packing LP constructed by $\mathrm{PLP}_{\text{dense}}(\boldsymbol{A}, \boldsymbol{b}, U)$ or $\mathrm{PLP}_{\text{sparse}}(\boldsymbol{A}, \boldsymbol{b}, U)$ up to accuracy $\epsilon = \mathrm{poly}(\epsilon, \dim(\boldsymbol{A})^{-1}, \kappa(\boldsymbol{A})^{-1})$. This gives a linear system solver whose running time has $\mathrm{poly}(1/\epsilon)$-dependence. We use the standard technique *iterative refinement*, to reduce this $\mathrm{poly}(1/\epsilon)$-dependence to $\log(1/\epsilon)$-dependence. For completeness, we include a proof below.

Lemma 4.1.7. *If we have a solver for* $\mathrm{LSA}(\boldsymbol{A}, \boldsymbol{b}, 0.1)$ *that works for arbitrary* $\boldsymbol{b}$*, then we can obtain a solver for* $\mathrm{LSA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$ *by iterating it* $O(\log(1/\epsilon))$ *times.*

Proof. Consider the linear equation $\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^\top \boldsymbol{b}$. Let $\boldsymbol{M} = \boldsymbol{A}^\top \boldsymbol{A}$ and $\boldsymbol{p} = \boldsymbol{A}^\top \boldsymbol{b}$. We basically do iterative refinement involving the matrix $\boldsymbol{M}$. Note the desired solution is $\boldsymbol{x}^* = \boldsymbol{M}^{-1}\boldsymbol{p}$. (See Fact 2.2.4).

For simplicity let's consider a 2-step version. We start with

$$\boldsymbol{p}^{(0)} = \boldsymbol{p},$$

and the LSA solver produces $\boldsymbol{x}^{(0)}$ such that

$$\left\|\boldsymbol{x}^{(1)} - \boldsymbol{M}^{-1}\boldsymbol{p}^{(0)}\right\|_{\boldsymbol{M}} \leq 0.1 \left\|\boldsymbol{M}^{-1}\boldsymbol{p}^{(0)}\right\|_{\boldsymbol{M}}.$$

Then we define

$$\boldsymbol{p}^{(1)} := \boldsymbol{p}^{(0)} - \boldsymbol{M}\boldsymbol{x}^{(1)}.$$

Note that at each step, we also have

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{x}\right),$$

so forming $\boldsymbol{b}$ by

$$\boldsymbol{b}^{(t+1)} = \left(\boldsymbol{b}^{(t)} - \boldsymbol{A}\boldsymbol{x}^{(t)}\right)$$

also works.

By definition we have

$$\boldsymbol{M}^{-1}\boldsymbol{p}^{(1)} = \boldsymbol{M}\boldsymbol{p}^{(0)} - \boldsymbol{x}^{(1)},$$

and thus

$$\left\|\boldsymbol{M}^{-1}\boldsymbol{p}^{(1)}\right\|_{\boldsymbol{M}} \leq 0.1\left\|\boldsymbol{M}^{-1}\boldsymbol{p}^{(0)}\right\|_{\boldsymbol{M}}.$$

Feeding in $\boldsymbol{p}^{(1)}$ as input to the solver in turn gives $\boldsymbol{x}^{(2)}$ such that

$$\left\|\boldsymbol{x}^{(2)} - \boldsymbol{M}^{-1}\boldsymbol{p}^{(1)}\right\|_{\boldsymbol{M}} \leq 0.1\left\|\boldsymbol{M}^{-1}\boldsymbol{p}^{(1)}\right\|_{\boldsymbol{M}} \leq 0.01\left\|\boldsymbol{M}^{-1}\boldsymbol{p}^{(0)}\right\|_{\boldsymbol{M}}.$$

Expanding the LHS gives

$$\boldsymbol{x}^{(2)} - \boldsymbol{M}^{-1}\boldsymbol{p}^{(1)} = \boldsymbol{x}^{(2)} + \boldsymbol{x}^{(1)} - \boldsymbol{M}^{-1}\boldsymbol{p},$$

which means $\boldsymbol{x}^{(1)} + \boldsymbol{x}^{(2)}$ is now a solution with error 0.01. Repeating this gives a $10\times$ smaller error after each step. □

Combining the Lemmas above, we proved Theorem 4.0.7 and Theorem 4.0.8.

CHAPTER 5

3-D TRUSS LINEAR SYSTEM SOLVER

In this chapter, we prove Theorem 1.1.4 and Theorem 1.1.5.

5.1 Main Algorithm and Proofs of Main Results

The key idea of our algorithm is to combine Nested Dissection and preconditioned iterative algorithms. Motivated by Nested Dissection and the concept r-division [77], we partition a convex edge-simple 3-D truss (see Definition 2.4.5) into small and separate pieces. Each piece has roughly the same number of vertices, and each piece has a well-shaped boundary consisting of a small number of vertices.

The bottleneck of Nested Dissection is the process of applying Gaussian elimination to the few top-level separators, after eliminating all interior vertices at lower levels (see Section 2.5 for a detailed explanation of nested dissection). At this step, one runs Gaussian elimination on the the Schur complement onto the top-level separators, which is usually dense.

To speed up Nested Dissection, we adapt the idea from support theory. We observe that, after eliminating all interior vertices, the Schur complement onto the boundaries can be well preconditioned by the boundaries alone. Instead of directly running Gaussian elimination on the Schur complement, we run an iterative algorithm Preconditioned Conjugate Gradient (PCG) to solve a sequence of sparse linear systems in the boundaries.

The phenomenon that the boundary itself is a good preconditioner for the Schur complement onto the boundary has a natural interpretation based on structural mechanics. The quadratic form associated with the Schur complement corresponds to the energy associated with deforming the whole truss by squishing or stretching the boundary vertices while leaving the interior intact and finding the positions of the interior vertices that minimize the

overall energy. We show that this energy is not much more than the energy that arises from applying the same deformation to just the boundary tetrahedrons after deleting the interior vertices.

Our main algorithm is presented in Algorithm 12. Its input consists of an edge-simple 3D truss, a column vector, an error parameter, an aspect ratio threshold, and a hollowing rate. Algorithm 12 will use the following subroutines: Algorithm 13, presented in Section 5.4, to compute hollowings of small aspect ratio truss components in line 6; Algorithm 15, presented in Section 5.5, for nested dissection of a partial state of of the preconditioner in line 8.

Algorithm 12 TRUSSSOLVER($\mathcal{T} = \langle V, \{\boldsymbol{p}_i\}_{i\in V}, T, E, \gamma\rangle, \boldsymbol{f}, \epsilon, c_\alpha, c_r$)

Input: a 3D truss $\mathcal{T} = \langle V, \{\boldsymbol{p}_i\}_{i\in V}, T, E, \gamma\rangle$ with n vertices, which is a union of k convex edge-simple trusses $\mathcal{T}_1, \ldots, \mathcal{T}_k$, a vector $\boldsymbol{f} \in \mathbb{R}^{3n}$, an error parameter $\epsilon > 0$. Constants for aspect ratio threshold $0 < c_\alpha < 1$, and hollowing rate $0 < c_r < 1$

Output: an approximate solution $\boldsymbol{x}$ such that $\|\boldsymbol{A}_\mathcal{T}\boldsymbol{x} - \boldsymbol{f}\|_2 \leq \epsilon \|\boldsymbol{f}\|_2$.

1: **for** each i **do**
2: Compute a bounding box B_i of $\mathcal{T}_i$.
3: **end for**
4: Let $\mathcal{I} = \{1 \leq i \leq k : \alpha(\mathcal{T}_i) \leq n_i^{c_\alpha}\}$.
5: **for** each $i \in \mathcal{I}$ **do**
6: Hollow out the interior vertices of $\mathcal{T}_i$ with parameter $r_i = n_i^{c_r}$ to form $\mathcal{H}_i$.
7: **end for**
8: Run nested dissection on the preconditioner (possibly with a specific set of separators).
9: Run preconditioned conjugate gradient with this preconditioner to solve the overall system.
10: **return** the solution $\boldsymbol{x}$.

Bounding eigenvalues of an edge-simple and stiffly connected truss. The key to analyze the running time of Algorithm 12 is to bound the relative condition number of the intermediate matrix and its preconditioner. By Theorem 2.5.3, this condition number determines the number of PCG iterations in line 9.

Since each vertex of a 3-D truss is incident to a constant number of edges with constant edge weights, it is not hard to check that its largest eigenvalue is some constant. The

following Lemma lower bounds its smallest non-zero eigenvalue.

Lemma 5.1.1. *Let $\mathcal{T}$ be an edge-simple and stiffly-connected 3D truss. Let n be the number of vertices of $\mathcal{T}$ and Δ be the diameter. Let $\boldsymbol{M}$ denote the associated stiffness matrix. Then, $\lambda_{\min}(\boldsymbol{M}) = \Omega(n^{-1}\Delta^{-4})$ and rank$(\boldsymbol{M}) = 3n - 6$.*

The proof of Lemma 5.1.1 is quite involved, which is in Section 5.2 and 5.3. This proof is motivated by the Path Lemma in [44]. It bounds the smallest non-zero eigenvalue of a 2-D triangle path. In their proof, they center / normalize a vector in the eigen-space w.r.t. the first triangle of the path. Then they show that the minimum energy for stretching or squeezing each triangle to some position propagates along the triangle path, at a rate polynomial in the length of the path. However, this does not hold for 3-D tetrahedron surfaces. The propagated energy changes exponentially. We deal with this by trying different centerings, and show there always exists a centering desirable for our purpose.

We remark that we think the bound given in Lemma 5.1.1 is not tight, from the observations of Matlab experiments. It is not clear whether one can derive Cheeger type inequalities for truss stiffness matrices (even for the simplest 2-D case), similar to the one in [46].

Proving the main result for small aspect ratio truss unions. Motivated by the r-division introduced by Frederickson [77], we divide each small aspect ratio truss component into smaller *regions*. Given the convexity of each truss component, we can guarantee that that the boundary of each small region is a well-shaped tetrahedron surface, and thus can be used as a good preconditioner of the Schur complement onto that boundary. The union of all these boundaries is called a *hollowing*. To create a hollowing, we fix two parameters: a bounding box B that determines the directions of each smaller chunks of the hollowing, and a size parameter r that controls the size of the smaller regions.

Definition 5.1.2 ((B, r)-hollowing)**.** *Given a convex edge-simple 3D truss $\mathcal{T} = \langle V, \{\boldsymbol{p}_i\}_{i\in V}, T, E, \gamma\rangle$, a bounding box B of $\mathcal{T}$, and a parameter r, a* (B, r)-hollowing *of $\mathcal{T}$ is another edge-simple*

3D truss $\mathcal{H} = \langle U, \{\boldsymbol{p}_i\}_{i \in U}, S, F, \gamma' \rangle$ *such that,* $U \subseteq V$*,* $S \subseteq T$*, and* F *is the subset of* E *that arises from edges in* S*, while* γ' *is just the restriction of* γ *to* F*. I.e. edges maintain the same stiffness factors as in* $\mathcal{T}$*. Also*

1. $\mathcal{H}$ *contains* $O(nr^{-1/3})$ *points.* $\mathcal{T} \setminus \mathcal{H}$ *consists of* $O(nr^{-1})$ *separate chunks, each of which has* $O(r)$ *vertices and is incident to* $O(r^{2/3})$ *vertices of* $\mathcal{H}$*.*

2. *for* every *plane* P *whose normal vector has angle* $\theta \in (0, \pi/2)$ *with the longest direction of* B*, the number of tetrahedrons in* $\mathcal{H}$ *intersected by* P *is*

$$O\left(n^{2/3}\alpha^{-1/3}r^{-1/3}\cos^{-2}\theta\right),$$

   where α *is the aspect ratio of* $\mathcal{T}$*.*

3. $\boldsymbol{A}_{\mathcal{H}} \preceq \textsc{Sc}[\boldsymbol{A}_{\mathcal{T}}]_U \preceq O(r^2)\boldsymbol{A}_{\mathcal{H}}$*.*

The next lemma describes the performance of algorithm HOLLOW, Algorithm 13 in Section 5.4, that we use to compute a (B, r)-hollowing of a convex edge-simple truss.

Lemma 5.1.3. *Given a convex edge-simple 3-D truss* $\mathcal{T} = \langle V, \{\boldsymbol{p}_i\}_{i \in V}, T, E, \gamma \rangle$ *with* n *vertices, a bounding box* B *of* $\mathcal{T}$*, and a positive integer* r *such that the aspect ratio of* $\mathcal{T}$ *is at most* $\sqrt{n/r}$*, the algorithm* HOLLOW$(\mathcal{T}, B, r)$ *returns a* (B, r)*-hollowing* $\mathcal{H}$ *of* $\mathcal{T}$*, and runs in time* $O(n)$*.*

We now prove our main result for small aspect ratio truss unions.

Proof of Theorem 1.1.4. Let $\mathcal{T}$ be a edge-simple 3-D truss with n vertices, formed from a union of k convex edge-simple trusses, say $\mathcal{T}_1, \ldots, \mathcal{T}_k$, each with aspect ratio at most $O(1)$. For each $1 \leq i \leq k$, let n_i be the number of vertices of $\mathcal{T}_i$, and define $r_i \stackrel{\text{def}}{=} n_i^{1/2}$. In Algorithm 12, for each $\mathcal{T}_i$, we compute a (B_i, r_i)-hollowing, where B_i is a bounding box

of $\mathcal{T}_i$. By Lemma 2.4.2 and 5.1.3, the total running time here is $O(n)$. In each (B_i, r_i)-hollowing region, we eliminate its interior vertices in total time

$$O\left(\sum_i n_i r_i^{-1} \cdot r_i^2\right) = O\left(n^{3/2}\right).$$

The Schur complement onto the boundaries has

$$O\left(\sum_i n_i r_i^{-1} \cdot (r_i^{2/3})^2\right) = O\left(n^{7/6}\right)$$

nonzeros. We then run preconditioned conjugate gradient (PCG) to solve the linear system in the Schur complement by preconditioning it via the union of the (B_i, r_i)-hollowings, say $\mathcal{T}'$. Note by Jensen's inequality, $\mathcal{T}'$ has size

$$O\left(\sum_i n_i r_i^{-1/3}\right) = O\left(\sum_i n_i^{5/6}\right) = O\left(k^{1/6} n^{5/6}\right).$$

Before running PCG, we compute a Cholesky factorization of $\boldsymbol{A}_{\mathcal{T}'}$ by nested dissection. According to Theorem 2.5.1, the running time is $O(k^{1/3}n^{5/3})$, and the fill-in size is $O(k^{2/9}n^{10/9})$. By Definition 5.1.2, the condition number is $O(\max_i r_i^2) = O(n)$. According to Theorem 2.5.3, the number of PCG iterations is at most $O(n^{1/2}\log(1/\epsilon))$ to output a solution up to accuracy ϵ. In each PCG iteration, we do a matrix-vector multiplication with the Schur complement in time $O(n^{7/6})$, and solve a linear system in $\boldsymbol{A}_{\mathcal{T}'}$ in time $O(k^{2/9}n^{10/9})$. Thus the total running time is $O(k^{1/3}n^{5/3}\log(1/\epsilon))$. □

Proving the main result for all-aspect ratio truss unions. We extend our result to cover the case when the union of convex edge-simple trusses also include trusses with arbitrarily large aspect ratios. The extension is based on an observation: large aspect ratio simplicial complexes imply the existence of small size separators. The following lemma specifies this observation, which we will prove in Section 5.5.2.

Lemma 5.1.4. *Given a convex edge-simple 3D truss $\mathcal{T} = \langle V, \{\boldsymbol{p}_i\}_{i \in V}, T, E, \gamma \rangle$ with aspect ratio at least $\alpha > 0$, and its bounding box B. Let $\boldsymbol{d} \in \mathbb{R}^3$ be a unit vector along the longest direction of B, and let $\boldsymbol{g} \in \mathbb{R}^3$ be a unit vector with $\boldsymbol{d} \cdot \boldsymbol{g} > 0$. Then every plane orthogonal to $\boldsymbol{g}$ intersects at most $O(n^{2/3}\alpha^{-1/3}(\boldsymbol{d} \cdot \boldsymbol{g})^{-1})$ tetrahedrons.*

Combining this together with the existence of hollowings for small aspect ratio trusses, we get the following lemma. It speeds up nested dissection for a partial state of our Gaussian elimination, which is crucial to prove the main result for all-aspect ratio truss unions.

Lemma 5.1.5 (Combining Separators)**.** *Given a edge-simple 3D truss $\mathcal{T} = \langle V, \{\boldsymbol{p}_i\}_{i \in V}, T, E, \gamma \rangle$, which is a union of k convex edge-simple trusses with up to n vertices in total. There exists a randomized algorithm which returns with high probability a truss $\mathcal{T}'$ by selectively computing (B_i, r_i) hollowings of some of the pieces with parameter*

$$r_i \leq n_i^{1/3}$$

so that a complete elimination of $\mathcal{T}'$ has size $O(n^{23/18}k^{44/9})$, and takes time $O(n^{11/6}k^{22/3})$ to compute.

The algorithm that achieves Lemma 5.1.5 is Algorithm 15 CONVEXTRUSSUNIONND in Section 5.5. Given these lemmas, we can now sketch a proof of Theorem 1.1.5.

Proof of Theorem 1.1.5. We bound the running time of Algorithm 12 TRUSSSOLVER with the preconditioner and nested dissection constructed as per Lemma 5.1.5.

Since all the hollowings involve pieces with $r_i \leq n_i^{1/3}$, Definition 5.1.2 gives a bound of $O(n^{2/3})$ on the condition number, and in turn a bound of $O(n^{1/3}\log(1/\epsilon))$ on the number of PCG iterations via Theorem 2.5.3. Furthermore, similar to the proof of Theorem 1.1.4, the Schur complement of $\mathcal{T}$ onto the elements of $\mathcal{T}'$ has size $O(n^{10/9})$, and computing them by eliminating all interior vertices of our hollowings takes time $O(n^{4/3})$.

Thus, the total running time of Algorithm 12 is

$$O\left(n^{4/3} + n^{11/6}k^{22/3} + n^{1/3}\log(1/\epsilon) \cdot \left(n^{10/9} + n^{23/18}k^{44/9}\right)\right) = O\left(n^{11/6}k^{22/3}\log(1/\epsilon)\right).$$

□

5.2 Bounding the Smallest Nonzero Eigenvalue of a edge-simple Truss

In this section, we prove Lemma 5.1.1, which lower bounds the smallest nonzero eigenvalue of a edge-simple 3D truss. We restate Lemma 5.1.1 in the following.

Lemma 5.1.1. *Let $\mathcal{T}$ be an edge-simple and stiffly-connected 3D truss. Let n be the number of vertices of $\mathcal{T}$ and Δ be the diameter. Let $\boldsymbol{M}$ denote the associated stiffness matrix. Then, $\lambda_{\min}(\boldsymbol{M}) = \Omega(n^{-1}\Delta^{-4})$ and rank$(\boldsymbol{M}) = 3n - 6$.*

5.2.1 Main Ideas

The proof of Lemma 5.1.1 is an extension of the path support lemma by Daitch and Spielman [44]. That proof relies on recentering a vector $\boldsymbol{q}$, which is a unit vector orthogonal to the null space, with respect to the a single face by transforming it along the null space of $\boldsymbol{M}$.

Let $\mathcal{T} = \langle V, \{\boldsymbol{p}_i\}_{i\in V}, T, E, \gamma\rangle$ be a 3D stiffly-connected truss over n vertices. The null space of the stiffness matrix of $\mathcal{T}$ can be characterized as:

1. $\boldsymbol{p}^x$, $\boldsymbol{p}^y$, $\boldsymbol{p}^z \in \mathbb{R}^{3n}$: for each $1 \leq i \leq n$, the corresponding 3-dimensional vector $\boldsymbol{p}_i^x$ ($\boldsymbol{p}_i^y$ and $\boldsymbol{p}_i^z$) has 1 for its x-coordinate (y-coordinate, and z-coordinate, respectively) and 0 for the other two coordinates.

2. $\boldsymbol{p}^{\perp xy}, \boldsymbol{p}^{\perp xz}, \boldsymbol{p}^{\perp yz} \in \mathbb{R}^{3n}$: fix an arbitrary index $1 \leq c \leq n$, for each $1 \leq i \leq n$:

$$\begin{aligned}
\boldsymbol{p}_i^{\perp xy} &= \left[-\left(\boldsymbol{p}_i - \boldsymbol{p}_c\right)_y, \left(\boldsymbol{p}_i - \boldsymbol{p}_c\right)_x, 0\right]^\top, \\
\boldsymbol{p}_i^{\perp xz} &= \left[-\left(\boldsymbol{p}_i - \boldsymbol{p}_c\right)_z, 0, \left(\boldsymbol{p}_i - \boldsymbol{p}_c\right)_x\right]^\top, \\
\boldsymbol{p}_i^{\perp yz} &= \left[0, -\left(\boldsymbol{p}_i - \boldsymbol{p}_c\right)_z, \left(\boldsymbol{p}_i - \boldsymbol{p}_c\right)_y\right]^\top.
\end{aligned}$$

Also, as many of our arguments are symmetric across dimensions, we will use d, d_1 and d_2 to represent symmetric indexing over the dimensions, or pairs of dimensions respectively. Finally, as centering and exploring a simplicial complex from a particular triangle introduces an ordering on the tetrahedrons, faces, and edges, we will define our edges, triangles, and tetrahedrons as ordered tuples:

1. Edges: $e = \langle e_1, e_2 \rangle$,
2. Triangles: we denote these as $s = \langle s_1, s_2, s_3 \rangle$. Here $e(s)$ means the edge $\langle s_1, s_2 \rangle$.
3. Tetrahedrons: an ordered 4-tuples of pairwise adjacent points, $t = \langle t_1, t_2, t_3, t_4 \rangle$. Here $s(t)$ means the (triangle) surface $\langle t_1, t_2, t_3 \rangle$, and $e(t)$ means the edge $e(s) = \langle t_1, t_2 \rangle$.

With these notations in mind, we can center a vector $\boldsymbol{q}$ w.r.t. a particular (oriented) triangle surface s. This can be viewed as an extension of the centering lemma in [44]. We prove the following Lemma in Section 5.2.2.

Lemma 5.2.1. *Given a stiffly-connected, edge-simple truss $\mathcal{T} = \langle V, \{\boldsymbol{p}_i\}_{i \in V}, T, E, \gamma \rangle$, let $\boldsymbol{q}$ be a vector orthogonal to the null space of its stiffness matrix. For each oriented triangle s, there exists a (unique) vector $\overline{\boldsymbol{q}}^{\langle s \rangle}$ with scalar shift parameters $c^{\langle s \rangle \perp xy}$, $c^{\langle s \rangle \perp xz}$, $c^{\langle s \rangle \perp yz}$:*

$$\overline{\boldsymbol{q}}^{\langle s \rangle} = \boldsymbol{q} + \sum_{d1d2} c^{\langle s \rangle \perp d1d2} \boldsymbol{p}^{\perp d1d2} \tag{5.1}$$

satisfying:

1. *the plane containing the points $\overline{\boldsymbol{q}}_s^{\langle s \rangle}$ is parallel to the plane containing $\boldsymbol{p}_s$.*

2. *The edge $\overline{\boldsymbol{q}}_{e(s)}^{\langle s \rangle}$ is parallel to the edge $\boldsymbol{p}_{e(s)}$.*

Daitch and Spielman then showed that with any centering, the value of $\boldsymbol{q}^T \boldsymbol{M} \boldsymbol{q}$ is lower bounded by the sum of a series of shifted values, or in simpler terms, the norm of $\overline{\boldsymbol{q}}_i^{\langle s \rangle} - \overline{\boldsymbol{q}}_j^{\langle s \rangle}$ for some edge ij. However, our extension of this bound (which we will describe next in Lemma 5.2.4) to the 3-D case has an exponential dependency on the distance in tetrahedrons between ij and s. As a result, we first show the existence of a good centering, namely one where there exist an edge close to s whose endpoints are far apart. This notion of distance can be defined in terms of 'hop count' of tetrahedrons.

Definition 5.2.2. *The* tetrahedron-distance *between a pair of objects x and y in a simplicial complex is the shortest sequence of tetrahedrons*

$$t^{(0)}, t^{(1)}, \ldots t^{(d)}$$

such that $x \subseteq t^{(0)}$, $y \subseteq t^{(d)}$, and for all $1 \leq i \leq d$, $t^{(i-1)}$ and $t^{(i)}$ share a triangle face.

We remark that because all edges and angles are within some constant range, this combinatorial distance is within constant factors of the Euclidean distance of the associated points. However, we will not make use of this connection.

Lemma 5.2.3. *Given a stiffly-connected, edge-simple truss $\mathcal{T}$ with n vertices and diameter Δ, let $\boldsymbol{q}$ be a unit vector orthogonal to the null space of the stiffness matrix of $\mathcal{T}$. There exists an oriented triangle s and a pair of points i, j within tetrahedron-distance $O(1)$ of s satisfying:*

$$\left\| \overline{\boldsymbol{q}}_i^{\langle s \rangle} - \overline{\boldsymbol{q}}_j^{\langle s \rangle} \right\|_2^2 = \Omega\left(\frac{1}{\Delta^4 n}\right).$$

We prove this lemma in Section 5.2.3.

We can then check, via an argument similar to [44], that such a centering and distance pair implies a large quadratic form. The following lemma will be proved in Section 5.3.

Lemma 5.2.4. *Given a stiffly-connected, edge-simple truss with stiffness matrix $\boldsymbol{M}$, an oriented triangle s, and a pair of points i, j within tetrahedron-distance h of s, we have*

$$\left\|\overline{\boldsymbol{q}}^{\langle s\rangle}\right\|_{\boldsymbol{M}}^2 \geq 2^{-\Theta(h)} \left\|\overline{\boldsymbol{q}}_i^{\langle s\rangle} - \overline{\boldsymbol{q}}_j^{\langle s\rangle}\right\|_2^2.$$

Proof of Lemma 5.1.1. Consider an arbitrarily fixed unit vector $\boldsymbol{q}$ that's orthogonal to the null space of $\boldsymbol{M}$. Let s be the centering given by Lemma 5.2.3, and let i, j be a pair of points within a constant tetrahedron-distance of s. Lemma 5.2.4 gives

$$\left\|\overline{\boldsymbol{q}}^{\langle s\rangle}\right\|_{\boldsymbol{M}}^2 \geq \Omega(1) \left\|\overline{\boldsymbol{q}}_i^{\langle s\rangle} - \overline{\boldsymbol{q}}_j^{\langle s\rangle}\right\|_2^2 \geq \Omega\left(\frac{1}{\Delta^4 n}\right). \tag{5.2}$$

On the other hand, by Equation (5.1), $\left\|\overline{\boldsymbol{q}}^{\langle s\rangle}\right\|_2 \geq \|\boldsymbol{q}\|_2 = 1$. Thus, $\lambda_{\min}(\boldsymbol{M}) \geq \Omega\left(\frac{1}{\Delta^4 n}\right)$. □

5.2.2 Centering a Vector (Proof of Lemma 5.2.1)

To prove Lemma 5.2.1, we define the following operation. Let P be any fixed plane in $\mathbb{R}^3$ and let $\boldsymbol{y} \in \mathbb{R}^3$. We define $\boldsymbol{y}^{\perp_P}$ to be the vector obtained by first projecting $\boldsymbol{y}$ onto plane P and then rotating the projected vector on the plane counterclockwise by $\pi/2$. The following claim shows that $\boldsymbol{y}^{\perp_P}$ can be written as a linear combination of $\boldsymbol{y}^{\perp_{xy}}, \boldsymbol{y}^{\perp_{yz}}, \boldsymbol{y}^{\perp_{xz}}$.

Claim 5.2.5 (Rotation matrix)**.** *Let P be any fixed plane in $\mathbb{R}^3$, and let $\boldsymbol{w}$ be its normal vector. Then,*

$$\forall \boldsymbol{y} \in \mathbb{R}^3, \quad \boldsymbol{y}^{\perp_P} = \boldsymbol{w}_z \boldsymbol{y}^{\perp_{xy}} - \boldsymbol{w}_y \boldsymbol{y}^{\perp_{xz}} + \boldsymbol{w}_x \boldsymbol{y}^{\perp_{yz}}.$$

Proof. Let

$$\boldsymbol{R} \stackrel{\text{def}}{=} \begin{pmatrix} 0 & -\boldsymbol{w}_z & \boldsymbol{w}_y \\ \boldsymbol{w}_z & 0 & -\boldsymbol{w}_x \\ -\boldsymbol{w}_y & \boldsymbol{w}_x & 0 \end{pmatrix}$$

be the rotation matrix which rotates a vector on the plane P counterclockwise by $\pi/2$. Then,

$$\boldsymbol{y}^{\perp_P} = \boldsymbol{R}\left(\boldsymbol{y} - (\boldsymbol{y}^\top \boldsymbol{w})\boldsymbol{w}\right) = \boldsymbol{R}\left(\boldsymbol{I} - \boldsymbol{w}^\top \boldsymbol{w}\right)\boldsymbol{y}.$$

We can check that $\boldsymbol{R}\boldsymbol{w} = \boldsymbol{0}$. It implies that $\boldsymbol{y}^{\perp_P} = \boldsymbol{R}\boldsymbol{y}$.

On the other hand,

$$\boldsymbol{w}_z \boldsymbol{y}^{\perp_{xy}} - \boldsymbol{w}_y \boldsymbol{y}^{\perp_{xz}} + \boldsymbol{w}_x \boldsymbol{y}^{\perp_{yz}} = \begin{pmatrix} 0 & -\boldsymbol{w}_z & \boldsymbol{w}_y \\ \boldsymbol{w}_z & 0 & -\boldsymbol{w}_x \\ -\boldsymbol{w}_y & \boldsymbol{w}_x & 0 \end{pmatrix} \boldsymbol{y} = \boldsymbol{R}\boldsymbol{y}.$$

Thus, the claim holds. □

Claim 5.2.6. *Let $\boldsymbol{h} \in \mathbb{R}^3$ be a nonzero vector. Then matrix*

$$\boldsymbol{H} \stackrel{\text{def}}{=} \begin{pmatrix} 0 & -\boldsymbol{h}_z & \boldsymbol{h}_y \\ \boldsymbol{h}_z & 0 & -\boldsymbol{h}_x \\ -\boldsymbol{h}_y & \boldsymbol{h}_x & 0 \\ \boldsymbol{h}_x & \boldsymbol{h}_y & \boldsymbol{h}_z \end{pmatrix}.$$

has rank 3.

Proof. The 2-by-2 bottom left submatrix has determinant $-(\boldsymbol{h}_y^2 + \boldsymbol{h}_x^2)$. If $\boldsymbol{h}_y^2 + \boldsymbol{h}_x^2 = 0$, then clearly $\boldsymbol{H}$ has rank 3 and we have done; otherwise, the 3rd and the 4th rows are independent.

Now it suffices to show that the 2nd row is independent of the 3rd and the 4th rows. Assume by contradiction, suppose

$$\begin{aligned} \boldsymbol{h}_z &= -\alpha \boldsymbol{h}_y + \beta \boldsymbol{h}_x, \\ 0 &= \alpha \boldsymbol{h}_x + \beta \boldsymbol{h}_y, \\ -\boldsymbol{h}_x &= \beta \boldsymbol{h}_z. \end{aligned}$$

By solving the last two equations, we get $\beta = -\boldsymbol{h}_x/\boldsymbol{h}_z$ and $\alpha = \boldsymbol{h}_y/\boldsymbol{h}_z$. Plugging these values into the 1st equation, we have $\boldsymbol{h}_x^2 + \boldsymbol{h}_y^2 + \boldsymbol{h}_z^2 = 0$, which contradicts that $\boldsymbol{h} \neq \boldsymbol{0}$. $\square$

Proof of Lemma 5.2.1. Let $\boldsymbol{w}$ be the normal vector of the plane containing $s = \langle \boldsymbol{p}_{i_1}, \boldsymbol{p}_{i_2}, \boldsymbol{p}_{i_3} \rangle$, that is,

$$\boldsymbol{w}^\top (\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_1}) = 0, \boldsymbol{w}^\top (\boldsymbol{p}_{i_3} - \boldsymbol{p}_{i_1}) = 0, \text{ and } \|\boldsymbol{w}\|_2 = 1.$$

We first show that there exist $\alpha_{xy}, \alpha_{xz}, \alpha_{yz} \in \mathbb{R}$ such that the vector

$$\boldsymbol{z} \stackrel{\text{def}}{=} \boldsymbol{q} + \sum_{d1d2} \alpha_{d1d2} \boldsymbol{p}^{\perp_{d1d2}}$$

satisfies the first condition. It suffices to show that the following linear system has a solution for real numbers α_{d1d2}'s:

$$\begin{aligned}
\boldsymbol{w}^\top (\boldsymbol{q}_{i_2} - \boldsymbol{q}_{i_1} + \sum_{d1d2} \alpha_{d1d2} (\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_1})^{\perp_{d1d2}}) = 0, \\
\boldsymbol{w}^\top (\boldsymbol{q}_{i_3} - \boldsymbol{q}_{i_1} + \sum_{d1d2} \alpha_{d1d2} (\boldsymbol{p}_{i_3} - \boldsymbol{p}_{i_1})^{\perp_{d1d2}}) = 0.
\end{aligned}$$

Rearrange it and write it in matrix form:

$$\begin{pmatrix}
\boldsymbol{w}^\top (\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_1})^{\perp_{xy}} & \boldsymbol{w}^\top (\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_1})^{\perp_{yz}} & \boldsymbol{w}^\top (\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_1})^{\perp_{xz}} \\
\boldsymbol{w}^\top (\boldsymbol{p}_{i_3} - \boldsymbol{p}_{i_1})^{\perp_{xy}} & \boldsymbol{w}^\top (\boldsymbol{p}_{i_3} - \boldsymbol{p}_{i_1})^{\perp_{yz}} & \boldsymbol{w}^\top (\boldsymbol{p}_{i_3} - \boldsymbol{p}_{i_1})^{\perp_{xz}}
\end{pmatrix}
\begin{pmatrix}
\alpha_{xy} \\ \alpha_{yz} \\ \alpha_{xz}
\end{pmatrix}
= \begin{pmatrix}
-\boldsymbol{w}^\top (\boldsymbol{x}_{i_2} - \boldsymbol{x}_{i_1}) \\
-\boldsymbol{w}^\top (\boldsymbol{x}_{i_3} - \boldsymbol{x}_{i_1})
\end{pmatrix}.$$

It suffices to show that the coefficient matrix has rank 2.

Assume by contradiction, there is some $k \in \mathbb{R}$ such that

$$\boldsymbol{w}^\top (\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_1})^{\perp_{d1d2}} = k \boldsymbol{w}^\top (\boldsymbol{p}_{i_3} - \boldsymbol{p}_{i_1})^{\perp_{d1d2}}, \quad \forall d1, d2$$

It equals to

$$\boldsymbol{w}^\top \left((\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_1}) - k(\boldsymbol{p}_{i_3} - \boldsymbol{p}_{i_1})\right)^{\perp_{d1d2}} = 0, \quad \forall d1, d2$$

Let $\boldsymbol{h} \stackrel{\text{def}}{=} (\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_1}) - k(\boldsymbol{p}_{i_3} - \boldsymbol{p}_{i_1})$. Since vectors $\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_1}, \boldsymbol{p}_{i_3} - \boldsymbol{p}_{i_1}$ are not parallel, we have $\boldsymbol{h} \neq \boldsymbol{0}$. Besides, $\boldsymbol{h} \perp \boldsymbol{w}$. Write the above equation in matrix form:

$$\begin{pmatrix} 0 & -\boldsymbol{h}_z & \boldsymbol{h}_y \\ \boldsymbol{h}_z & 0 & -\boldsymbol{h}_x \\ -\boldsymbol{h}_y & \boldsymbol{h}_x & 0 \\ \boldsymbol{h}_x & \boldsymbol{h}_y & \boldsymbol{h}_z \end{pmatrix} \boldsymbol{w} = \boldsymbol{0}.$$

By Claim 5.2.6, the coefficient matrix has rank 3, which implies that $\boldsymbol{w} = \boldsymbol{0}$. It contradicts that $\|\boldsymbol{w}\|_2 = 1$.

Then we show that there exist $\beta_{xy}, \beta_{xz}, \beta_{yz} \in \mathbb{R}$ such that

$$\boldsymbol{q}^{\langle s \rangle} = \boldsymbol{z} + \sum_{d1d2} \beta_{d1d2} \boldsymbol{p}^{\perp_{d1d2}}$$

satisfies both conditions.

Let P be the plane containing $\boldsymbol{p}_{i_1}, \boldsymbol{p}_{i_2}, \boldsymbol{p}_{i_3}$. Let $\boldsymbol{g} \in \mathbb{R}^{3n}$ satisfy

$$\boldsymbol{g}_i = (\boldsymbol{p}_i - \boldsymbol{p}_1)^{\perp_P}, \quad \forall 1 \leq i \leq n$$

Then, there exists an appropriate multiplier $\gamma \in \mathbb{R}$ such that the vector

$$(\boldsymbol{z}_{i_2} - \boldsymbol{z}_{i_1}) + \gamma(\boldsymbol{g}_{i_2} - \boldsymbol{g}_{i_1}) = (\boldsymbol{z}_{i_2} - \boldsymbol{z}_{i_1}) + \gamma(\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_1})^{\perp_P}$$

is parallel to $\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_1}$. Besides, since both $\boldsymbol{z}$ and $\boldsymbol{g}$ are parallel to the plane P, the vector $\boldsymbol{q}^{\langle s \rangle} = \boldsymbol{z} + \gamma \boldsymbol{g}$ is parallel to the plane P.

By Claim 5.2.5, there exists real numbers $\beta_{xy}, \beta_{xz}, \beta_{yz}$ such that

$$\gamma \boldsymbol{g} = \sum_{d1d2} \beta_{d1d2} \boldsymbol{p}^{\perp_{d1d2}}.$$

Let

$$c^{\langle s \rangle \perp_{d1d2}} = \alpha_{d1d2} + \beta_{d1d2}, \quad \forall d1, d2$$

This completes the proof. □

5.2.3 Existence of Good Centering (Proof of Lemma 5.2.3)

In this section we prove Lemma 5.2.3.

The proof is by contradiction. We assume that for *every* centering at a triangle s, for every pair $\langle i, j \rangle$ within tetrahedron-distance 3 of $\langle s \rangle$ satisfies

$$\left\| \overline{\boldsymbol{q}}_i^{\langle s \rangle} - \overline{\boldsymbol{q}}_j^{\langle s \rangle} \right\|_2 \leq \epsilon, \tag{5.3}$$

where $\epsilon \stackrel{\text{def}}{=} \sqrt{\frac{1}{\beta \Delta^4 n}}$ for a sufficiently large constant β.

Recall that in Lemma 5.2.1, for each centering at $\langle s \rangle$, we define 3 scalar coefficients

$$c^{\langle s \rangle \perp xy}, c^{\langle s \rangle \perp xz}, c^{\langle s \rangle \perp yz} \in \mathbb{R}$$

for the 3 null space vectors in Equation (5.1). We will write the vector containing these 3 coefficients as $\boldsymbol{c}^{\langle s \rangle}$.

To prove Lemma 5.2.3, we need the following lemma. It says that, under the assumption in Equation (5.3), the difference between the coefficient vectors w.r.t. to two close centering triangles is small.

Lemma 5.2.7. *Assume that for every centering triangle s and every pair of points i, j within distance 3 of $\langle s \rangle$ satisfies Equation* (5.3)*. Then for every pair of centering triangles*

s_1 and s_2 within constant tetrahedron distance to each other, we have

$$\left\| \boldsymbol{c}^{\langle s_1 \rangle} - \boldsymbol{c}^{\langle s_2 \rangle} \right\|_2 = O(\epsilon).$$

The next lemma implies that: for any two vertices such that each is centered w.r.t. a triangle close to itself, the difference between the two centered vertices is small.

Lemma 5.2.8. *Let u, w be two arbitrary vertices of $\mathcal{T}$. Let s_u (s_w) be a triangle containing u (and w, respectively). Under the assumption in Equation* (5.3)*, we have*

$$\left\| \overline{\boldsymbol{q}}_u^{\langle s_u \rangle} - \overline{\boldsymbol{q}}_w^{\langle s_w \rangle} \right\|_2 = O\left(\Delta^2 \epsilon\right).$$

Proof. Let $u = v_1, v_2, \ldots, v_f = w$ be a shortest path from u to w. Note $f \leq \Delta$. Let s_i be a triangle next to v_i for $2 \leq i \leq f-1$. The path from vertex u, centered at $\langle s_u \rangle$, to vertex w, centered at $\langle s_w \rangle$, can be expressed as the following:

$$\overline{\boldsymbol{q}}_u^{\langle s_u \rangle} - \overline{\boldsymbol{q}}_w^{\langle s_w \rangle} = \overline{\boldsymbol{q}}_u^{\langle s_u \rangle} - \overline{\boldsymbol{q}}_{v_2}^{\langle s_u \rangle} + \sum_{2 \leq i \leq f-1} \left(\overline{\boldsymbol{q}}_{v_i}^{\langle s_i \rangle} - \overline{\boldsymbol{q}}_{v_i}^{\langle s_{i+1} \rangle} \right) + \sum_{2 \leq i \leq f-1} \left(\overline{\boldsymbol{q}}_{v_i}^{\langle s_{i+1} \rangle} - \overline{\boldsymbol{q}}_{v_{i+1}}^{\langle s_{i+1} \rangle} \right).$$

Taking ℓ_2 norm on both sides and applying the triangle inequality, we an bound the norm of the LHS by the sum of ℓ_2 norm of each term in the RHS.

By Equation (5.1) and the triangle inequality,

$$\left\| \overline{\boldsymbol{q}}_{v_i}^{\langle s_i \rangle} - \overline{\boldsymbol{q}}_{v_i}^{\langle s_{i+1} \rangle} \right\|_2 \leq \sum_{d_1 d_2} \left| \boldsymbol{c}^{\langle s_i \rangle \perp d1 d2} - \boldsymbol{c}^{\langle s_{i+1} \rangle \perp d1 d2} \right| \left\| \left(\boldsymbol{p}_{v_i} - \boldsymbol{p}_c \right)^{\perp d1 d2} \right\|_2.$$

Apply Lemma 5.2.7:

$$\left\| \overline{\boldsymbol{q}}_{v_i}^{\langle s_i \rangle} - \overline{\boldsymbol{q}}_{v_i}^{\langle s_{i+1} \rangle} \right\|_2 = O(\Delta \epsilon). \tag{5.4}$$

Together with our assumption in Equation (5.3), we have

$$\left\|\overline{\boldsymbol{q}}_u^{\langle s_u\rangle} - \overline{\boldsymbol{q}}_w^{\langle s_w\rangle}\right\|_2 \leq O(\Delta^2\epsilon).$$

□

Now we prove Lemma 5.2.3.

Proof of Lemma 5.2.3. For each vertex i, let s_i denote an arbitrary triangle next to vertex i. We can write vector $\boldsymbol{q}$ as

$$\boldsymbol{q} = \widehat{\boldsymbol{q}} + \tilde{\boldsymbol{q}} - \sum_{d_1 d_2} c^{\langle s_1\rangle \perp d_1 d_2} \boldsymbol{p}^{\perp d_1 d_2} + \boldsymbol{e}, \tag{5.5}$$

where

$$\widehat{\boldsymbol{q}}_i = \overline{\boldsymbol{q}}_i^{\langle s_i\rangle} - \overline{\boldsymbol{q}}_1^{\langle s_1\rangle}, \tilde{\boldsymbol{q}}_i = \overline{\boldsymbol{q}}_i^{\langle s_1\rangle} - \overline{\boldsymbol{q}}_i^{\langle s_i\rangle}, \boldsymbol{e}_i = \overline{\boldsymbol{q}}_1^{\langle s_1\rangle}.$$

Note that the last two terms of Equation (5.5) are in the null space of $\boldsymbol{M}$. Thus,

$$\|\widehat{\boldsymbol{q}} + \tilde{\boldsymbol{q}}\|_2 \geq \|\boldsymbol{q}\|_2 = 1.$$

On the other hand, by the triangle inequality,

$$\|\widehat{\boldsymbol{q}} + \tilde{\boldsymbol{q}}\|_2 \leq \|\widehat{\boldsymbol{q}}\|_2 + \|\tilde{\boldsymbol{q}}\|_2 = \left(\sum_i \left\|\overline{\boldsymbol{q}}_i^{\langle s_i\rangle} - \overline{\boldsymbol{q}}_1^{\langle s_1\rangle}\right\|_2^2\right)^{1/2} + \left(\sum_i \left\|\overline{\boldsymbol{q}}_i^{\langle s_1\rangle} - \overline{\boldsymbol{q}}_i^{\langle s_i\rangle}\right\|_2^2\right)^{1/2}.$$

We apply Lemma 5.2.8 to the first term, and apply Equation (5.4) (which is true for any two close centering triangles) Δ times for the second term:

$$\|\widehat{\boldsymbol{q}} + \tilde{\boldsymbol{q}}\|_2 \leq \|\widehat{\boldsymbol{q}}\|_2 + \|\tilde{\boldsymbol{q}}\|_2 = O(\sqrt{n}\Delta^2\epsilon).$$

By our choice of $\epsilon \leftarrow \frac{1}{\beta\sqrt{n}\Delta^2}$ for a sufficiently large constant β, we get a contradiction. □

It remains to prove Lemma 5.2.7. For it, we define a matrix w.r.t. a given vector $\boldsymbol{v} \in \mathbb{R}^3$:

$$\boldsymbol{Q}_{\boldsymbol{v}} \stackrel{\text{def}}{=} \left(\boldsymbol{v}^{\perp yz}, \boldsymbol{v}^{\perp xz}, \boldsymbol{v}^{\perp xy}\right) = \begin{pmatrix} 0 & -\boldsymbol{v}_z & \boldsymbol{v}_y \\ \boldsymbol{v}_z & 0 & -\boldsymbol{v}_x \\ -\boldsymbol{v}_y & \boldsymbol{v}_x & 0 \end{pmatrix}. \tag{5.6}$$

We will use the following properties of $\boldsymbol{Q}_{\boldsymbol{v}}$.

Lemma 5.2.9. *For the matrix $\boldsymbol{Q}_{\boldsymbol{v}}$ as defined in Equation* (5.6)*, its singular values are* $0, 1, 1$*, and its null space is multiples of the vector $\boldsymbol{v}$.*

Proof. For any vector $\boldsymbol{p} \in \mathbb{R}^3$, the cross product $\boldsymbol{v} \times \boldsymbol{p} = \boldsymbol{Q}_{\boldsymbol{v}}\boldsymbol{p}$. That is, $\boldsymbol{Q}_{\boldsymbol{v}}\boldsymbol{p}$ is the vector obtained by:

1. First projecting $\boldsymbol{p}$ onto the plane with normal vector $\boldsymbol{v}$, say $P_{\perp \boldsymbol{v}}$, and
2. then rotating the projected vector on the plane $P_{\perp \boldsymbol{v}}$ by $\frac{\pi}{2}$ counterclockwise.

From this description, we can infer that

$$\mathrm{null}(\boldsymbol{Q}_{\boldsymbol{v}}) = \textbf{Span}\,(\boldsymbol{v})$$

and any vector orthogonal to $\mathrm{null}(\boldsymbol{Q}_{\boldsymbol{v}})$ is on this plane $P_{\perp \boldsymbol{v}}$. Such vectors are not affected by the first step projection, and their lengths are not changed by the subsequent rotation. This means for such vectors $\boldsymbol{u} \perp \mathrm{null}(\boldsymbol{Q}_{\boldsymbol{v}})$ we have $\boldsymbol{u}^\top \boldsymbol{Q}_{\boldsymbol{v}}^\top \boldsymbol{Q}_{\boldsymbol{v}} \boldsymbol{u} = \|\boldsymbol{u}\|_2^2$. Thus, the singular values of $\boldsymbol{Q}_{\boldsymbol{v}}$ are 1,1,0. □

Lemma 5.2.10. *If $\boldsymbol{v}$ and $\boldsymbol{u}$ are vectors with length at least* 1 *such that the angle between them is $\theta \in (0, \pi)$, then the matrix*

$$\boldsymbol{Q}_{\boldsymbol{v}}^\top \boldsymbol{Q}_{\boldsymbol{v}} + \boldsymbol{Q}_{\boldsymbol{u}}^\top \boldsymbol{Q}_{\boldsymbol{u}}$$

is full rank, and has minimum eigenvalue at least $2\sin^2 \frac{\theta}{2}$.

Proof. Let $\boldsymbol{h} \in \mathbb{R}^3$ be a unit vector. Decompose $\boldsymbol{h}$:

$$\boldsymbol{h} = \alpha_v \boldsymbol{v} + \beta_v \hat{\boldsymbol{v}} = \alpha_u \boldsymbol{u} + \beta_u \hat{\boldsymbol{u}},$$

where $\alpha_v, \alpha_u, \beta_v, \beta_u \in \mathbb{R}$, $\hat{\boldsymbol{v}}$ is a unit vector orthogonal to $\boldsymbol{v}$, and $\hat{\boldsymbol{u}}$ is a unit vector orthogonal to $\boldsymbol{u}$. Then,

$$\boldsymbol{h}^\top (\boldsymbol{Q}_v^\top \boldsymbol{Q}_v + \boldsymbol{Q}_u^\top \boldsymbol{Q}_u) \boldsymbol{h} = \beta_v^2 \left\| \boldsymbol{Q}_v \hat{\boldsymbol{v}} \right\|_2^2 + \beta_u^2 \left\| \boldsymbol{Q}_u \hat{\boldsymbol{u}} \right\|_2^2 .$$

By Lemma 5.2.9, $\left\| \boldsymbol{Q}_v \hat{\boldsymbol{v}} \right\|_2 = 1$ and $\left\| \boldsymbol{Q}_u \hat{\boldsymbol{u}} \right\|_2 = 1$. Thus,

$$\boldsymbol{h}^\top (\boldsymbol{Q}_v^\top \boldsymbol{Q}_v + \boldsymbol{Q}_u^\top \boldsymbol{Q}_u) \boldsymbol{h} = \beta_v^2 + \beta_u^2 = 2 - (\alpha_v^2 + \alpha_u^2).$$

The second equality is due to that $\boldsymbol{h}$ is a unit vector.

$$\alpha_v^2 + \alpha_u^2 = \left\| \boldsymbol{h}^\top \boldsymbol{v} \right\|_2^2 + \left\| \boldsymbol{h}^\top \boldsymbol{u} \right\|_2^2 \leq 2 \cos^2 \frac{\theta}{2}.$$

Therefore,

$$\lambda_{\min}(\boldsymbol{Q}_v^\top \boldsymbol{Q}_v + \boldsymbol{Q}_u^\top \boldsymbol{Q}_u) \geq 2 \sin^2 \frac{\theta}{2}.$$

$\theta < \pi$ implies that $\lambda_{\min}(\boldsymbol{Q}_v^\top \boldsymbol{Q}_v + \boldsymbol{Q}_u^\top \boldsymbol{Q}_u) > 0$, and thus the matrix $\boldsymbol{Q}_v^\top \boldsymbol{Q}_v + \boldsymbol{Q}_u^\top \boldsymbol{Q}_u$ has full rank. □

Equipped with the above lemmas, we prove Lemma 5.2.7.

Proof of Lemma 5.2.7. Let s_1 and s_2 be two triangle centerings for which there exist two edges belong to same tetrahedron, say $(i, j), (j, k)$, such that vertices i, j, k are all within tetrahedron-distance constant h of both s_1 and s_2.

Recall $\overline{\boldsymbol{q}}^{\langle s_1 \rangle}$ is defined in Equation (5.1). Subtracting

$$\overline{\boldsymbol{q}}_i^{\langle s_1 \rangle} = \boldsymbol{q}_i + \sum_{d_1 d_2} c^{\langle s_1 \rangle \perp d_1 d_2} \left(\boldsymbol{p}_i^{\perp d_1 d_2} - \boldsymbol{p}_c^{\perp d_1 d_2} \right)$$

from the corresponding equation for j gives:

$$\overline{\boldsymbol{q}}_i^{\langle s_1 \rangle} - \overline{\boldsymbol{q}}_j^{\langle s_1 \rangle} = \boldsymbol{q}_i - \boldsymbol{q}_j + \sum_{d_1 d_2} c^{\langle s_1 \rangle \perp d_1 d_2} \left(\boldsymbol{p}_i^{\perp d_1 d_2} - \boldsymbol{p}_j^{\perp d_1 d_2} \right).$$

Plugging in the definition of $\boldsymbol{Q}_{(\boldsymbol{p}_i - \boldsymbol{p}_j)}$:

$$\overline{\boldsymbol{q}}_i^{\langle s_1 \rangle} - \overline{\boldsymbol{q}}_j^{\langle s_1 \rangle} = \boldsymbol{q}_i - \boldsymbol{q}_j + \boldsymbol{Q}_{(\boldsymbol{p}_i - \boldsymbol{p}_j)} \boldsymbol{c}^{\langle s_1 \rangle}.$$

Subtracting this equation from centering triangle s_2 in turn cancels the $\boldsymbol{q}_i - \boldsymbol{q}_j$ term on the RHS, giving:

$$\boldsymbol{Q}_{(\boldsymbol{p}_i - \boldsymbol{p}_j)} \left(\boldsymbol{c}^{\langle s_1 \rangle} - \boldsymbol{c}^{\langle s_2 \rangle} \right) = \left(\overline{\boldsymbol{q}}_i^{\langle s_1 \rangle} - \overline{\boldsymbol{q}}_j^{\langle s_1 \rangle} \right) - \left(\overline{\boldsymbol{q}}_i^{\langle s_2 \rangle} - \overline{\boldsymbol{q}}_j^{\langle s_2 \rangle} \right).$$

Together with the corresponding equation for j, k,

$$\begin{pmatrix} \boldsymbol{Q}_{(\boldsymbol{p}_i - \boldsymbol{p}_j)} \\ \boldsymbol{Q}_{(\boldsymbol{p}_j - \boldsymbol{p}_k)} \end{pmatrix} \left(\boldsymbol{c}^{\langle s_1 \rangle} - \boldsymbol{c}^{\langle s_2 \rangle} \right) = \begin{pmatrix} \left(\overline{\boldsymbol{q}}_i^{\langle s_1 \rangle} - \overline{\boldsymbol{q}}_j^{\langle s_1 \rangle} \right) - \left(\overline{\boldsymbol{q}}_i^{\langle s_2 \rangle} - \overline{\boldsymbol{q}}_j^{\langle s_2 \rangle} \right) \\ \left(\overline{\boldsymbol{q}}_j^{\langle s_1 \rangle} - \overline{\boldsymbol{q}}_k^{\langle s_1 \rangle} \right) - \left(\overline{\boldsymbol{q}}_j^{\langle s_2 \rangle} - \overline{\boldsymbol{q}}_k^{\langle s_2 \rangle} \right) \end{pmatrix}.$$

Multiplying $\begin{pmatrix} \boldsymbol{Q}_{(\boldsymbol{p}_i - \boldsymbol{p}_j)}^\top & \boldsymbol{Q}_{(\boldsymbol{p}_j - \boldsymbol{p}_k)}^\top \end{pmatrix}$ on both sides gives:

$$\begin{aligned} &\left(\boldsymbol{Q}_{(\boldsymbol{p}_i - \boldsymbol{p}_j)}^\top \boldsymbol{Q}_{(\boldsymbol{p}_i - \boldsymbol{p}_j)} + \boldsymbol{Q}_{(\boldsymbol{p}_j - \boldsymbol{p}_k)}^\top \boldsymbol{Q}_{(\boldsymbol{p}_j - \boldsymbol{p}_k)} \right) \left(\boldsymbol{c}^{\langle s_1 \rangle} - \boldsymbol{c}^{\langle s_2 \rangle} \right) \\ &\qquad = \begin{pmatrix} \boldsymbol{Q}_{(\boldsymbol{p}_i - \boldsymbol{p}_j)}^\top & \boldsymbol{Q}_{(\boldsymbol{p}_j - \boldsymbol{p}_k)}^\top \end{pmatrix} \begin{pmatrix} \left(\overline{\boldsymbol{q}}_i^{\langle s_1 \rangle} - \overline{\boldsymbol{q}}_j^{\langle s_1 \rangle} \right) - \left(\overline{\boldsymbol{q}}_i^{\langle s_2 \rangle} - \overline{\boldsymbol{q}}_j^{\langle s_2 \rangle} \right) \\ \left(\overline{\boldsymbol{q}}_j^{\langle s_1 \rangle} - \overline{\boldsymbol{q}}_k^{\langle s_1 \rangle} \right) - \left(\overline{\boldsymbol{q}}_j^{\langle s_2 \rangle} - \overline{\boldsymbol{q}}_k^{\langle s_2 \rangle} \right) \end{pmatrix}. \end{aligned}$$

Solving the above linear equations and taking norm on both sides:

$$\left\|\boldsymbol{c}^{\langle s_1\rangle}-\boldsymbol{c}^{\langle s_2\rangle}\right\|_2 \leq \left\|\left(\boldsymbol{Q}_{(\boldsymbol{p}_i-\boldsymbol{p}_j)}^\top \boldsymbol{Q}_{(\boldsymbol{p}_i-\boldsymbol{p}_j)}+\boldsymbol{Q}_{(\boldsymbol{p}_j-\boldsymbol{p}_k)}^\top \boldsymbol{Q}_{(\boldsymbol{p}_j-\boldsymbol{p}_k)}\right)^{-1}\right\|_2 \cdot \left\|\begin{pmatrix} \boldsymbol{Q}_{(\boldsymbol{p}_i-\boldsymbol{p}_j)}^\top & \boldsymbol{Q}_{(\boldsymbol{p}_j-\boldsymbol{p}_k)}^\top \end{pmatrix}\right\|_2 \left\|\begin{pmatrix} \left(\overline{\boldsymbol{q}}_i^{\langle s_1\rangle}-\overline{\boldsymbol{q}}_j^{\langle s_1\rangle}\right)-\left(\overline{\boldsymbol{q}}_i^{\langle s_2\rangle}-\overline{\boldsymbol{q}}_j^{\langle s_2\rangle}\right) \\ \left(\overline{\boldsymbol{q}}_j^{\langle s_1\rangle}-\overline{\boldsymbol{q}}_k^{\langle s_1\rangle}\right)-\left(\overline{\boldsymbol{q}}_j^{\langle s_2\rangle}-\overline{\boldsymbol{q}}_k^{\langle s_2\rangle}\right) \end{pmatrix}\right\|_2.$$

Applying Lemma 5.2.9 and 5.2.10 on the first two terms, and applying the triangle inequality and the assumption in Equation (5.3) give:

$$\left\|\boldsymbol{c}^{\langle s_1\rangle}-\boldsymbol{c}^{\langle s_2\rangle}\right\|_2 \leq O(\epsilon).$$

□

5.3 Proof of Path Lemma

We now prove Lemma 5.2.4, which lower bounds the quadratic form of an edge-simple and stiffly-connected truss stiffness matrix by the distance between two centered points. As we now only deal with a single centering in this section, we will drop this superscription for simplicity and relabel all indices w.r.t. this centering. Our relabeling is similar to that in [44].

5.3.1 Relabeling Tetrahedrons and Vertices

Fix an arbitrary tetrahedron, say t_1, and we center the vector $\boldsymbol{q}$ w.r.t. one of the triangle faces of t_1 as in Lemma 5.2.1. Use t_1 as root, we run breadth-first-search (BFS) in the rigidity graph (refer to Definition 2.4.1) of $\mathcal{T}$, and relabel the tetrahedrons of $\mathcal{T}$ according to this BFS ordering. For example, the neighbor tetrahedrons of t_1 are labeled as $t_2, t_3, \ldots$. Let T_{BFS} be the corresponding BFS tree in which each node represents a tetrahedron in $\mathcal{T}$.

Based on T_{BFS}, we relabel the vertices of $\mathcal{T}$ as follows. We label the vertices of t_1 by

$-2, -1, 0, 1$ in an arbitrary order. Each child of t_1 (and subsequent recursions) shares a triangle face with their respective parent tetrahedron and thus only requires us to label one vertex per child tetrahedron, which can be labeled according to the BFS ordering.

For each newly labeled vertex j, we use t_j to denote the tetrahedron encompassing the t_j and its parental face. Besides, we use a 3-dimensional vector σ_j to consist of the indexes of the parental face sorted in ascending order, and is assumed that indexing this vector is implicitly modulo 3. Then, $t_j = \{j, \sigma_j(1), \sigma_j(2), \sigma_j(3)\}$. For completeness, we define $\sigma_1 \stackrel{\text{def}}{=} \{-2, -1, 0\}$. See Figure 5.1 for an example.

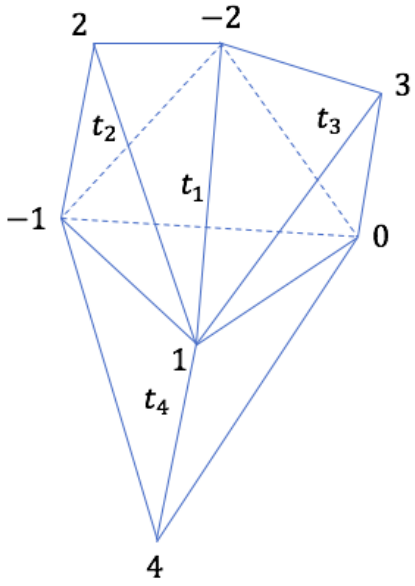

Figure 5.1: An example of the relabeling of tetrahedrons and vertices: $t_1 = \{-2, -1, 0, 1\}, t_2 = \{-2, -1, 1, 2\}, t_3 = \{-2, 0, 1, 3\}, t_4 = \{-1, 0, 1, 4\}$ and $\sigma_1 = \{-2, -1, 0\}, \sigma_2 = \{-2, -1, 1\}, \sigma_3 = \{-2, 1, 0\}, \sigma_4 = \{0, -1, 1\}$.

5.3.2 Distance between Local Minimizers and the Centered Vectors

Let $\boldsymbol{q} \in \mathbb{R}^{3n}$ be a unit vector which is orthogonal to $\textbf{Span}\left(\boldsymbol{p}^x, \boldsymbol{p}^y, \boldsymbol{p}^z, \boldsymbol{p}^{\perp xy}, \boldsymbol{p}^{\perp yz}, \boldsymbol{p}^{\perp xz}\right)$ defined at the beginning of Section 5.2.1. By Lemma 5.2.1, there exist scalars $c^{\langle t_1\rangle \perp xy}, c^{\langle t_1\rangle \perp xz}, c^{\langle t_1\rangle \perp yz} \in \mathbb{R}$ such that the vector

$$\overline{\boldsymbol{q}}^{\langle t_1\rangle} \stackrel{\text{def}}{=} \boldsymbol{q} + \sum_{d_1 d_2} c^{\langle t_1\rangle \perp d_1 d_2} \boldsymbol{p}^{\perp d_1 d_2}$$

satisfies:

1. the plane containing $\overline{\boldsymbol{q}}_{-2}^{\langle t_1\rangle}, \overline{\boldsymbol{q}}_{-1}^{\langle t_1\rangle}, \overline{\boldsymbol{q}}_{0}^{\langle t_1\rangle}$ is parallel to the plane σ_{t_1}, and

2. $\overline{\boldsymbol{q}}_{-2}^{\langle t_1\rangle} - \overline{\boldsymbol{q}}_{-1}^{\langle t_1\rangle}$ is parallel to $(\sigma_{t_1}(1) - \sigma_{t_1}(2))$.

We drop the superscription $\langle t_1\rangle$ when the context is clear.

For each $0 \leq i \leq n-3$, we define a 3-dimensional vector $\boldsymbol{y}_i$. $\boldsymbol{y}_0$ is a vector on the plane of $\sigma_{t_1} = \{\boldsymbol{p}_{-2}, \boldsymbol{p}_{-1}, \boldsymbol{p}_0\}$ and minimizes the energy / quadratic form, suppose both $\overline{\boldsymbol{q}}_{-2}^{\langle t_1\rangle}, \overline{\boldsymbol{q}}_{-1}^{\langle t_1\rangle}$ are fixed. That is,

$$\begin{aligned} &(\boldsymbol{p}_0 - \boldsymbol{p}_j)^\top (\boldsymbol{y}_0 - \overline{\boldsymbol{q}}_j) = 0, \quad \forall j \in \{-2, -1\} \\ &\boldsymbol{w}^\top (\boldsymbol{y}_0 - \overline{\boldsymbol{q}}_{-2}) = 0 \end{aligned} . \tag{5.7}$$

For each $1 \leq i \leq n-3$, $\boldsymbol{y}_i$ is a vector which minimizes the energy / quadratic form, suppose all the three points of σ_j are fixed. That is,

$$(\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)})^\top (\boldsymbol{y}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)}) = 0, \quad \forall j \in \{1, 2, 3\} \tag{5.8}$$

We then define the distance between each local minimizer $\boldsymbol{y}_i$ and the centered vector $\overline{\boldsymbol{q}}_i$. Specifically,

$$\boldsymbol{d}_i \stackrel{\text{def}}{=} \begin{cases} \overline{\boldsymbol{q}}_{-1} - \overline{\boldsymbol{q}}_{-2}, & i = -1 \\ \overline{\boldsymbol{q}}_i - \boldsymbol{y}_i, & 0 \leq i \leq n-3 \end{cases} \tag{5.9}$$

This definition intermediately gives that $\forall -1 \leq i \leq n-3, 1 \leq j \leq \min\{i+2, 3\}$,

$$\boldsymbol{d}_i^\top (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)}) = (\overline{\boldsymbol{q}}_i - \boldsymbol{y}_i)^\top (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)}) = (\overline{\boldsymbol{q}}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)})^\top (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)}). \tag{5.10}$$

Note that equation (5.10) describes the net stress on an edge.

We can explicitly express these $\boldsymbol{d}_i$'s by solving Equations (5.7) and (5.8).

Let

$$\boldsymbol{F}_0 \stackrel{\text{def}}{=} \begin{pmatrix} (\boldsymbol{p}_0 - \boldsymbol{p}_{-2})^\top \\ (\boldsymbol{p}_0 - \boldsymbol{p}_{-1})^\top \\ \boldsymbol{w}^\top \end{pmatrix}.$$

Let $\boldsymbol{h}_{-2}, \boldsymbol{h}_{-1}, \boldsymbol{h}_{\boldsymbol{w}} \in \mathbb{R}^3$ such that for each $1 \leq k \leq 3$,

$$\boldsymbol{h}_{-2}(k) \stackrel{\text{def}}{=} \det \begin{pmatrix} (\boldsymbol{p}_0 - \boldsymbol{p}_{-1})^{(k+1)_3} & (\boldsymbol{p}_0 - \boldsymbol{p}_{-1})^{(k+2)_3} \\ \boldsymbol{w}^{(k+1)_3} & \boldsymbol{w}^{(k+2)_3} \end{pmatrix},$$

$$\boldsymbol{h}_{-1}(k) \stackrel{\text{def}}{=} \det \begin{pmatrix} (\boldsymbol{p}_0 - \boldsymbol{p}_{-2})^{(k+1)_3} & (\boldsymbol{p}_0 - \boldsymbol{p}_{-2})^{(k+2)_3} \\ \boldsymbol{w}^{(k+1)_3} & \boldsymbol{w}^{(k+2)_3} \end{pmatrix},$$

$$\boldsymbol{h}_{\boldsymbol{w}}(k) \stackrel{\text{def}}{=} \det \begin{pmatrix} (\boldsymbol{p}_0 - \boldsymbol{p}_{-1})^{(k+1)_3} & (\boldsymbol{p}_0 - \boldsymbol{p}_{-1})^{(k+2)_3} \\ (\boldsymbol{p}_0 - \boldsymbol{p}_{-2})^{(k+1)_3} & (\boldsymbol{p}_0 - \boldsymbol{p}_{-2})^{(k+2)_3} \end{pmatrix}.$$

where $(k+1)_3$ and $(k+2)_3$ denotes accessing specific dimensions of vectors $\in \mathbb{R}^3$ modulo 3. Then we define

$$\boldsymbol{H}_{0,2} \stackrel{\text{def}}{=} \boldsymbol{h}_{-2}(\boldsymbol{p}_0 - \boldsymbol{p}_{-2})^\top, \quad \boldsymbol{H}_{0,1} \stackrel{\text{def}}{=} \boldsymbol{h}_{-1}(\boldsymbol{p}_0 - \boldsymbol{p}_{-1})^\top, \quad \text{and } \boldsymbol{H}_{\boldsymbol{w}} \stackrel{\text{def}}{=} \boldsymbol{h}_{\boldsymbol{w}} \boldsymbol{w}^\top.$$

Similarly, for $1 \leq i \leq n-3$, let

$$\boldsymbol{F}_i \stackrel{\text{def}}{=} \begin{pmatrix} (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(1)})^\top \\ (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(2)})^\top \\ (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(3)})^\top \end{pmatrix},$$

and

$$\boldsymbol{H}_{i,j} \stackrel{\text{def}}{=} \boldsymbol{h}_{i,j}(\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)})^\top, \quad \forall 1 \leq j \leq 3, \tag{5.11}$$

where $\boldsymbol{h}_{i,j} \in \mathbb{R}^3$ with

$$\boldsymbol{h}_{i,j}(k) \stackrel{\text{def}}{=} \det \begin{pmatrix} (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+1)_3)})^{(k+1)_3} & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+1)_3)})^{(k+2)_3} \\ (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+2)_3)})^{(k+1)_3} & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+2)_3)})^{(k+2)_3} \end{pmatrix}.$$

We can check that the following $\boldsymbol{y}_i$'s satisfy Equations (5.7) and (5.8):

$$\boldsymbol{y}_i = \begin{cases} \frac{1}{\det(\boldsymbol{F}_0)} \left((\boldsymbol{H}_{0,2} + \boldsymbol{H}_{\boldsymbol{w}}) \overline{\boldsymbol{q}}_{-2} + \boldsymbol{H}_{0,1} \overline{\boldsymbol{q}}_{-1} \right), & i = 0 \\ \frac{1}{\det(\boldsymbol{F}_i)} \sum_{1 \le j \le 3} \boldsymbol{H}_{i,j} \overline{\boldsymbol{q}}_{\sigma_i(j)}, & \forall 1 \le i \le n \end{cases}$$

Claim 5.3.1.

$$\det(\boldsymbol{F}_i) = (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)})^\top \boldsymbol{h}_{i,j}. \tag{5.12}$$

Proof. Let

$$\boldsymbol{Q} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

be a 3×3 matrix. Then, the determinant of $\boldsymbol{Q}$ is

$$\det(\boldsymbol{Q}) = a \cdot \det \left(\begin{pmatrix} e & f \\ h & i \end{pmatrix} \right) - b \cdot \det \left(\begin{pmatrix} d & f \\ g & i \end{pmatrix} \right) + c \cdot \det \left(\begin{pmatrix} d & e \\ g & h \end{pmatrix} \right).$$

Applying the above rule to $\boldsymbol{F}_i$ gives Eqaution (5.12). □

Claim 5.3.2.

$$\det(\boldsymbol{F}_i)\boldsymbol{I} = \sum_{1 \le j \le 3} \boldsymbol{H}_{i,j}, \quad \forall i \ge 0$$

Proof. We first show that the diagonals of $\sum_{j\in[3]} \boldsymbol{H}_{i,j}$ are equal to $\det(\boldsymbol{F}_i)$.

$$
\begin{aligned}
&\sum_{j\in[3]} \boldsymbol{H}_{i,j}(1,1) \\
=&\sum_{j\in[3]} (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)})(1) \cdot \det \begin{pmatrix} (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+1)_3)})(2) & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+1)_3)})(3) \\ (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+2)_3)})(2) & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+2)_3)})(3) \end{pmatrix} \\
=&\det \begin{pmatrix} (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(1)})^\top \\ (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(2)})^\top \\ (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(3)})^\top \end{pmatrix} \\
=&\det(\boldsymbol{F}_i).
\end{aligned}
$$

Similarly, we have

$$\sum_{j\in[3]} \boldsymbol{H}_{i,j}(2,2) = \sum_{j\in[3]} \boldsymbol{H}_{i,j}(3,3) = \det(\boldsymbol{F}_i).$$

Then, we show that the off-diagonals of $\sum_{j\in[3]} \boldsymbol{H}_{i,j}$ are 0.

$$
\begin{aligned}
&\sum_{j\in[3]} \boldsymbol{H}_{i,j}(1,2) \\
=&\sum_{j\in[3]} (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+1)_3)})(2) \cdot \det \begin{pmatrix} (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+1)_3)})(2) & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+1)_3)})(3) \\ (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+2)_3)})(2) & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i((j+2)_3)})(3) \end{pmatrix} \\
=&\det \begin{pmatrix} (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(1)})(2) & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(1)})(2) & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(1)})(3) \\ (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(2)})(2) & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(2)})(2) & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(2)})(3) \\ (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(3)})(2) & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(3)})(2) & (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(3)})(3) \end{pmatrix} \\
=&0.
\end{aligned}
$$

The last equation is due to the 3 columns of the matrix are linearly dependent. Similarly,

$$\sum_{j\in[3]} \boldsymbol{H}_{i,j}(s,t) = 0, \quad \forall 1 \le s \ne t \le 3.$$

This completes the proof. □

Plugging the above equations into the definition of $\boldsymbol{d}_i$'s gives:

Claim 5.3.3. *For each* $-1 \le i \le n-3$,

$$\boldsymbol{d}_i = \overline{\boldsymbol{q}}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)} + \frac{1}{\det(\boldsymbol{F}_i)} \sum_{\substack{1\le j'\le \min\{i+2,3\}\\ j'\ne j}} \boldsymbol{H}_{i,j'} \left(\overline{\boldsymbol{q}}_{\sigma_i(j)} - \overline{\boldsymbol{q}}_{\sigma_i(j')}\right).$$

5.3.3 Bounding the Norm of $\overline{\boldsymbol{q}}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)}$ in terms of $\boldsymbol{d}_i$'s

Rearranging the above equation gives that for each $-1 \le i \le n-3, 1 \le j \le \min\{i+2,3\}$,

$$\overline{\boldsymbol{q}}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)} = \boldsymbol{d}_i - \frac{1}{\det(\boldsymbol{F}_i)} \sum_{\substack{1\le j'\le \min\{i+2,3\}\\ j'\ne j}} \boldsymbol{H}_{i,j'} \left(\overline{\boldsymbol{q}}_{\sigma_i(j)} - \overline{\boldsymbol{q}}_{\sigma_i(j')}\right). \tag{5.13}$$

Our goal is to express each $\overline{\boldsymbol{q}}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)}$ as a function of $\boldsymbol{d}_i, \boldsymbol{d}_{i-1}, \ldots, \boldsymbol{d}_{-1}$. If each term $\overline{\boldsymbol{q}}_{\sigma_i(j)} - \overline{\boldsymbol{q}}_{\sigma_i(j')}$ in the right hand side of Equation (5.13) satisfies

$$\{\sigma_i(j), \sigma_i(j')\} = \{i_1, \sigma_{i_1}(j_1)\}$$

for some $i_1 < i$ and $1 \le j_1 \le \min\{i_1+2, 3\}$, then we can substitute it by Equation (5.13) with the left hand side being $\overline{\boldsymbol{q}}_{i_1} - \overline{\boldsymbol{q}}_{\sigma_{i_1}(j_1)}$. The substitution terminates when the right hand side term becomes $\pm\left(\overline{\boldsymbol{q}}_{-1} - \overline{\boldsymbol{q}}_{-2}\right) = \pm\boldsymbol{d}_{-1}$.

Claim 5.3.4. *Let* $\overline{\boldsymbol{q}}_{\sigma_{i_1}(j_1)} - \overline{\boldsymbol{q}}_{\sigma_{i_1}(j_2)}$ *(*$0 \le i_1 \le n-3, 1 \le j_1 \ne j_2 \le \min\{i+2,3\}$*) be a term appearing in the right hand side of Equation* (5.13)*. There exist* $-1 \le i_2 < i_1, 1 \le$

$j_3 \leq \min\{i_2 + 2, 3\}$ *satisfying*

$$\{\sigma_{i_1}(j_1), \sigma_{i_1}(j_2)\} = \{i_2, \sigma_{i_2}(j_3)\}.$$

Proof. Without the loss of generality, assume $\sigma_{i_1}(j_1) > \sigma_{i_1}(j_2)$. If $i_1 = 0$, then $\{\sigma_{i_1}(j_1), \sigma_{i_1}(j_2)\} = \{-1, \sigma_{-1}(1)\}$; if $i_1 = 1$, then $\{\sigma_{i_1}(j_1), \sigma_{i_1}(j_2)\} \in \{\{0, \sigma_0(1)\}, \{0, \sigma_0(2)\}, \{-1, \sigma_0(1)\}\}$. The remaining proof focuses on $i_1 \geq 2$.

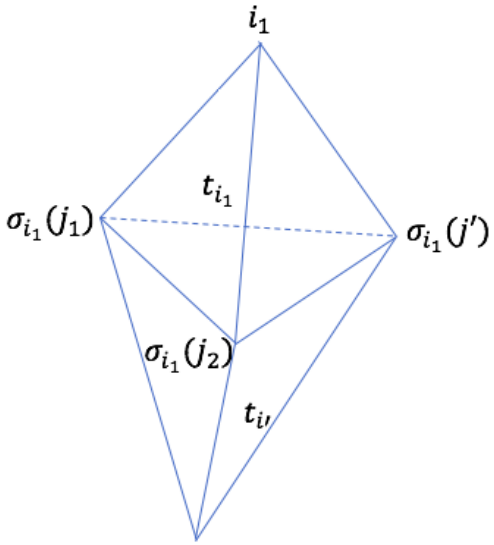

Figure 5.2: Tetrahedrons t_{i_1} and $t_{i'}$

Let $\sigma_{i_1}(j')$ be the other vertex in tetrahedron t_{i_1}. Let $t_{i'}$ be the parent of t_{i_1} in the BFS tree T_{BFS}. $i' < i_1$, and the two tetrahedrons t_{i_1} and $t_{i'}$ share a triangle face containing vertices $\sigma_{i_1}(j_1), \sigma_{i_1}(j_2), \sigma_{i_1}(j')$. See Figure 5.2. If $i' = 1$, then

$$\{\sigma_{i_1}(j_1), \sigma_{i_1}(j_2)\} \in \{\{i_2, \sigma_{i_2}(j_3)\} : -1 \leq i_2 \leq 1, 1 \leq j_3 \leq \min\{i_2 + 2, 3\}\}.$$

Otherwise $i' \geq 2$, we prove the statement by case analysis.

Case 1. $\sigma_{i_1}(j_1) = \max_{1 \leq k \leq 3}\{\sigma_{i_1}(k)\}$. By our labeling rules, vertex i' is the one with the

maximum index among all the vertices of $t_{i'}$ and it is contained in the triangle shared by t_{i_1} and $t_{i'}$. Thus, $\sigma_{i_1}(j_1) = i'$ and

$$\{\sigma_{i_1}(j_1), \sigma_{i_1}(j_2)\} = \{i', \sigma_{i'}(j_3)\}$$

for some $1 \leq j_3 \leq 3$.

Case 2. $\sigma_{i_1}(j') = \max_{1\leq k\leq 3}\{\sigma_{i_1}(k)\}$. Then, $\sigma_{i_1}(j') = i'$ and $\sigma_{i_1}(j_1), \sigma_{i_1}(j_2) \in \{\sigma_{i'}(k) : 1 \leq k \leq 3\}$. Note $i' < i$. By induction on the tetrahedron index i', we can see that there exist i_2, j_3 satisfying $\{\sigma_{i_1}(j_1), \sigma_{i_1}(j_2)\} = \{i_2, \sigma_{i_2}(j_3)\}$. □

We use a recursion tree $T_{\text{rec}}^{(i,j)}$ to express the process of recursively substituting $\overline{\boldsymbol{q}}_{i'} - \overline{\boldsymbol{q}}_{\sigma_{i'}(j')}$ via Equation (5.13). The root of $T_{\text{rec}}^{(i,j)}$ is $\overline{\boldsymbol{q}}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)}$, and each node u of $T_{\text{rec}}^{(i,j)}$ represents a term $\overline{\boldsymbol{q}}_{i_u} - \overline{\boldsymbol{q}}_{\sigma_{i_u}(j_u)}$ which is substituted at that point. The leaves are $\pm(\overline{\boldsymbol{q}}_{-1} - \overline{\boldsymbol{q}}_{-2})$, which equals $\pm\boldsymbol{d}_{-1}$ by Equation (5.9).

Claim 5.3.5. *The number of nodes in the recursion tree $T_{rec}^{(i,j)}$ is at most 2^i.*

Proof. Since $T_{\text{rec}}^{(i,j)}$ is a binary tree, it suffices to prove that the height of $T_{\text{rec}}^{(i,j)}$ is at most i. For each non-leaf node u of $T_{\text{rec}}^{(i,j)}$, let u_1 be a child of u. According to Equation (5.13) and our labeling rules, i_{u_1} is a vertex in tetrahedron t_{i_u} and $i_{u_1} < i_u$. Let t_k be the tetrahedron such that t_k and t_{i_u} share a triangle face containing vertices $\sigma_{i_u}(1), \sigma_{i_u}(2), \sigma_{i_u}(3)$. Then in T_{BFS}, t_k is the parent of t_{i_u}. Since $i_{u_1} < i_u$, in T_{BFS}, the depth of $t_{i_{u_1}}$ is smaller than the depth of t_{i_u}. See Figure 5.3. It implies that the height of $T_{\text{rec}}^{(i,j)}$ is at most the height of T_{BFS}, which is at most i. □

At each node u of the recursion tree $T_{\text{rec}}^{(i,j)}$, by applying Equation (5.13) with the left hand side being $\overline{\boldsymbol{q}}_{i_u} - \overline{\boldsymbol{q}}_{\sigma_{i_u}(j_u)}$, we introduce a term related to $\boldsymbol{d}_{i_u}$. For each non-root node u, denote $\text{pa}(u)$ the parent of u in $T_{\text{rec}}^{(i,j)}$. Let P_u be the path from the root node to node u in

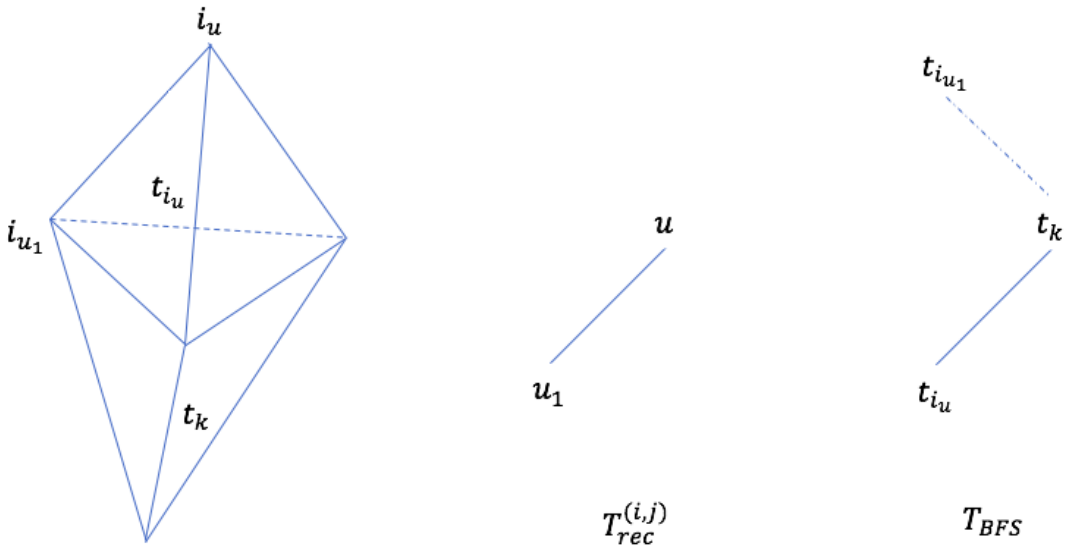

Figure 5.3: If u_1 is a child of u in $T_{\text{rec}}^{(i,j)}$, then the depth of $t_{i_{u_1}}$ is smaller than the depth of t_{i_u} in T_{BFS}.

$T_{\text{rec}}^{(i,j)}$. For each non-root node u, define

$$\boldsymbol{M}_u \stackrel{\text{def}}{=} \prod_{v \in P_{pa(u)}} \frac{\boldsymbol{H}_{i_v,j_v}}{\det(\boldsymbol{F}_{i_v})}. \tag{5.14}$$

For root node r, define $\boldsymbol{M}_r \stackrel{\text{def}}{=} \boldsymbol{I}$. By recursively applying Equation (5.13), we get

$$\overline{\boldsymbol{q}}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)} = \sum_{u \in T_{\text{rec}}^{(i,j)}} \text{sgn}(u) \boldsymbol{M}_u \boldsymbol{d}_{i_u},$$

where $\text{sgn}(u) \in \{+1, -1\}$. By the triangle inequality and the multiplicative inequality of 2-norm,

$$\left\| \overline{\boldsymbol{q}}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)} \right\|_2 \leq \sum_{u \in T_{\text{rec}}^{(i,j)}} \left\| \boldsymbol{M}_u \right\|_2 \left\| \boldsymbol{d}_{i_u} \right\|_2. \tag{5.15}$$

We bound $\left\| \boldsymbol{M}_u \right\|_2$ for each non-root node u by the following claim.

Claim 5.3.6. *Let $l_{\max}$ be the maximum edge length, and let $V_{\min}$ be the minimum tetrahedron volume. Define $c \stackrel{\text{def}}{=} \frac{l_{\max}^3}{6V_{\min}}$, which is a constant by our assumption that all tetrahedrons have constant aspect ratios and all edge lengths are constant. For each non-root u in $T_{rec}^{(i,j)}$, $\|\boldsymbol{M}_u\|_2 \leq c^i$.*

Proof. Let $i = u_1 > u_2 > \ldots > u_t > u_{t+1} = u$ be the vertices along the path P_u. By the proof of Claim 5.3.5, the length of P_u is at most i. Let $i_k = i_{u_k}$ and $j_k = j_{u_k}$ for each $1 \leq k \leq t$. By Equations (5.14), (5.11), and (5.12),

$$\begin{aligned}\boldsymbol{M}_u &= \prod_{i_1 \geq i_l \geq i_t} \frac{\boldsymbol{h}_{i_l,j_l}(\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(j_l)})^\top}{(\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(j_l)})^\top \boldsymbol{h}_{i_l,j_l}} \\ &= \boldsymbol{h}_{i_1,j_1} \left(\prod_{i_1 \geq i_l \geq i_{t-1}} \frac{(\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(j_l)})^\top \boldsymbol{h}_{(i_{l+1},j_{l+1})}}{(\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(j_l)})^\top \boldsymbol{h}_{(i_l,j_l)}} \right) \frac{(\boldsymbol{p}_{i_t} - \boldsymbol{p}_{\sigma_{i_l}(j_t)})^\top}{(\boldsymbol{p}_{i_t} - \boldsymbol{p}_{\sigma_{i_l}(j_t)})^\top \boldsymbol{h}_{(i_t,j_t)}}.\end{aligned}$$

Recall that from Claim 5.3.1,

$$(\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(j_l)})^\top \boldsymbol{h}_{(i_l,j_l)} = \det(\boldsymbol{F}_{i_l}),$$

which is equal to the volume of the tetrahedron generated by vectors $\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(1)}, \boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(2)}$, and $\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(3)}$. Similarly, $(\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(j_l)})^\top \boldsymbol{h}_{(i_{l+1},j_{l+1})}$ equals to the volume of the tetrahedron generated by vectors $\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(j_l)}, \boldsymbol{p}_{i_{l+1}} - \boldsymbol{p}_{\sigma_{i_{l+1}}((j_{l+1}+1)_3)}$, and $\boldsymbol{p}_{i_{l+1}} - \boldsymbol{p}_{\sigma_{i_{l+1}}((j_{l+1}+2)_3)}$. Thus, by our definition $c = l_{\max}^3/(6V_{\min}) \geq V_{\max}/V_{\min}$,

$$\left| \frac{(\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(j_l)})^\top \boldsymbol{h}_{(i_{l+1},j_{l+1})}}{(\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(j_l)})^\top \boldsymbol{h}^{(j_l)}_{(i_l,j_l)}} \right| \leq c.$$

Thus,

$$\|\boldsymbol{M}_u\|_2 \leq \left(\prod_{i_1 \geq i_l \geq i_{t-1}} \left| \frac{(\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(j_l)})^\top \boldsymbol{h}_{(i_{l+1},j_{l+1})}}{(\boldsymbol{p}_{i_l} - \boldsymbol{p}_{\sigma_{i_l}(j_l)})^\top \boldsymbol{h}_{(i_l,j_l)}} \right| \right) \left\| \frac{\boldsymbol{h}_{(i_1,j_l)}(\boldsymbol{p}_{i_t} - \boldsymbol{p}_{\sigma_{i_t}(j_t)})^\top}{(\boldsymbol{p}_{i_t} - \boldsymbol{p}_{\sigma_{i_t}(j_t)})^\top \boldsymbol{h}_{(i_t,j_t)}} \right\|_2 = O(c^i).$$

□

Applying Claim 5.3.6 to Equation (5.15),

$$\left\|\overline{\boldsymbol{q}}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)}\right\|_2 = O\left(c^i \sum_{u \in T_{\text{rec}}^{(i,j)}} \|\boldsymbol{d}_{i_u}\|_2\right).$$

By Claim 5.3.5 and the fact that $-1 \leq i_u \leq i$ for each $u \in T_{\text{rec}}^{(i,j)}$,

$$\left\|\overline{\boldsymbol{q}}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)}\right\|_2 = O\left((2c)^i \sum_{-1 \leq i' \leq i} \|\boldsymbol{d}_{i'}\|_2\right). \tag{5.16}$$

5.3.4 Bounding the Quadratic Form $(\overline{\boldsymbol{q}})^\top \boldsymbol{M} \overline{\boldsymbol{q}}$

Note

$$(\overline{\boldsymbol{q}})^\top \boldsymbol{M} \overline{\boldsymbol{q}} \geq \sum_{\substack{-1 \leq i \leq n-3 \\ 1 \leq j \leq \min\{i+2,3\}}} \left((\overline{\boldsymbol{q}}_i - \overline{\boldsymbol{q}}_{\sigma_i(j)})^\top (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)})\right)^2.$$

By Equation (5.10),

$$(\overline{\boldsymbol{q}})^\top \boldsymbol{M} \overline{\boldsymbol{q}} \geq \sum_{\substack{-1 \leq i \leq n-3 \\ 1 \leq j \leq \min\{i+2,3\}}} \left(\boldsymbol{d}_i^\top (\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)})\right)^2.$$

By the fact that $\boldsymbol{d}_{-1} = \overline{\boldsymbol{q}}_{-1} - \overline{\boldsymbol{q}}_{-2}$ is parallel to $\boldsymbol{p}_{-1} - \boldsymbol{p}_{-2}$,

$$\left(\boldsymbol{d}_{-1}^\top (\boldsymbol{p}_{-1} - \boldsymbol{p}_{-2})\right)^2 = \Omega\left(\|\boldsymbol{d}_{-1}\|_2^2\right).$$

Claim 5.3.7 (Lemma 3.7 of [44])**.** *Under the assumption: the angle between $\boldsymbol{p}_0 - \boldsymbol{p}_{-1}$ and $\boldsymbol{p}_0 - \boldsymbol{p}_{-2}$ is in the range $[\theta, \pi - \theta]$ for some constant θ. For any fixed unit vector $\boldsymbol{y} \in \mathbb{R}^3$,*

$$\sum_{j \in \{-2,-1\}} \left(\boldsymbol{y}^\top (\boldsymbol{p}_0 - \boldsymbol{p}_j)\right)^2 = \Theta(1).$$

Claim 5.3.8. *Under the assumption of constant edge lengths: for each $1 \leq i \leq n-3$, the*

determinant of the matrix:

$$\det\left(\begin{pmatrix} \boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(1)} & \boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(2)} & \boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(3)} \end{pmatrix}\right) = \Theta(1).$$

Furthermore, for any fixed unit vector $\boldsymbol{y} \in \mathbb{R}^3$*, for each* $1 \leq i \leq n-3$*,*

$$\sum_{1 \leq j \leq 3} \left(\boldsymbol{y}^\top \left(\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)}\right)\right)^2 = \Theta(1).$$

Proof. The determinant of matrix $\begin{pmatrix} \boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(1)} & \boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(2)} & \boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(3)} \end{pmatrix}$ is the signed volume of tetrahedron $t_i = \{i, \sigma_i(1), \sigma_i(2), \sigma_i(3)\}$, which is constant by our assumption.

We claim that for any unit vector $\boldsymbol{y}$, there exists some $j \in \{1,2,3\}$ such that $\boldsymbol{y}^\top(\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)}) = \Theta(1)$. Assume by contradiction, for every $j \in \{1,2,3\}$ we have $\boldsymbol{y}^\top(\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)}) = o(1)$. This means that all three vectors $\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(1)}, \boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(2)}, \boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(3)}$ are between two 2D planes with distance $o(1)$ and orthogonal to $\boldsymbol{y}$. This contradicts the assumption that the volume of tetrahedron $t_i = \{i, \sigma_i(1), \sigma_i(2), \sigma_i(3)\}$ is constant.

Thus, we have

$$\sum_{1 \leq j \leq 3} \left(\boldsymbol{y}^\top \left(\boldsymbol{p}_i - \boldsymbol{p}_{\sigma_i(j)}\right)\right)^2 = \Theta(1).$$

□

Therefore,

$$(\overline{\boldsymbol{q}})^\top \boldsymbol{M} \overline{\boldsymbol{q}} = \Omega\left(\sum_{-1 \leq i \leq n} \|\boldsymbol{d}_i\|_2^2\right).$$

Applying the Cauchy-Schwarz inequality on the above equation and Equation (5.16) gives the path lemma.

5.4 Computing a (B, r)-Hollowing of a Convex Edge-simple Truss

In this section, we present Algorithm 13 HOLLOW, which is used as a subroutine for line 6 of Algorithm 12 TRUSSSOLVER. The input of Algorithm 13 consists of a convex edge-simple 3D truss $\mathcal{T} = \langle \{\boldsymbol{p}_i\}_{i \in V}, T, E, \gamma \rangle$, a bounding box B and an integer parameter r. The algorithm outputs a (B, r)-hollowing $\mathcal{H}$ of $\mathcal{T}$. We can check that the algorithm terminates in time $O(|V|)$. These together prove Lemma 5.1.3.

Algorithm 13 HOLLOW($\mathcal{T} = \langle \{\boldsymbol{p}_i\}_{i \in V}, T, E, \gamma \rangle, B, r$)

Input: a convex edge-simple 3D truss $\mathcal{T} = \langle \{\boldsymbol{p}_i\}_{i \in V}, T, E, \gamma \rangle$, a bounding box B, a positive integer $r \leq n/\alpha(\mathcal{T})^2$

Output: a (B, r)-hollowing of $\mathcal{T}$

1: Compute r-division planes, which are planes orthogonal to each of the three directions of B such that these planes divide B into $O(n/r)$ small cubes of side length $O(r^{1/3})$ each.
2: $\mathcal{H} \leftarrow$ tetrahedrons intersecting an r-division plane.
3: $\mathcal{H} \leftarrow \mathcal{H} \cup$ tetrahedrons on the boundary of $\mathcal{T}$.
4: Make $\mathcal{H}$ stiffly-connected by adding a minimal number of tetrahedrons in $\mathcal{T}$.
5: **return** $\mathcal{H}$.

Given that each individual tetrahedron has constant volume and constant aspect ratio, the following observation converts counting the number of tetrahedrons in a truss intersecting a 2D plane into the intersection area of this truss and the plane.

Observation 5.4.1. *Let $\mathcal{T}$ be a convex edge-simple 3D truss, and let P be a 2D plane. Let $A(\mathcal{T} \cap P)$ be the intersection area of $\mathcal{T}$ and P. Then the number of tetrahedrons in $\mathcal{T}$ intersecting P is upper bounded by $O(\max\{A(\mathcal{T} \cap P), 1\})$.*

Proof. Since every individual tetrahedron in $\mathcal{T}$ has constant volume and constant aspect ratio, a tetrahedron of $\mathcal{T}$ intersects P only if all points of this tetrahedron is within some constant distance of P. The number of tetrahedrons of $\mathcal{T}$ intersecting P can be upper bounded by the volume within some constant distance to $\mathcal{T} \cap P$. Thus, the number of tetrahedrons of $\mathcal{T}$ intersecting P is at most $O(\max\{A(\mathcal{T} \cap P), 1\})$. □

5.4.1 Bounding the Size of $\mathcal{H}$

In this section, we show that $\mathcal{H} = \text{HOLLOW}(\mathcal{T}, B, r)$, computed by Algorithm 13, has a small size. That is, $\mathcal{H}$ satisfies the first condition of the (B, r)-hollowing definition in Definition 5.1.2.

Note in Algorithm 13 line 1, the bounding box B is divided into $O(n/r)$ small cubes of volume $O(r)$ each. We call a small cube as a *region*. The tetrahedrons in a region are the tetrahedrons of $\mathcal{H}$ which intersect a single small cube. A tetrahedron can appear in at most eight regions.

The following lemma upper bounds the number of tetrahedrons of $\mathcal{H}$ in each region.

Lemma 5.4.2. *Given a convex edge-simple 3D truss $\mathcal{T}$ of n vertices, a bounding box B and a positive integer $r \leq n/\alpha(\mathcal{T})^2$, let $\mathcal{H} = \text{HOLLOW}(\mathcal{T}, B, r)$ returned by Algorithm 13. Then, $\mathcal{H}$ has at most $O(nr^{-1/3})$ tetrahedrons.*

Note the shortest side length of the bounding box of $\mathcal{T}$ is at least $n^{1/3}\alpha^{-2/3}$. The requirement $r \leq n/\alpha(\mathcal{T})^2$ guarantees that the shortest side length is at least $r^{1/3}$ so that an r-division exists.

Proof. Note $\mathcal{H}$ has $O(n/r)$ regions. It suffices to show that each region of $\mathcal{H}$ has at most $O(r^{2/3})$ tetrahedrons.

Let R be a region of $\mathcal{H}$. A tetrahedron of $\mathcal{H}$ belongs to region R if either this tetrahedron is within constant distance to the boundary of R, or this tetrahedron is within constant distance to the part of the boundary of $\mathcal{T}$ that's contained in R. Since every tetrahedron of $\mathcal{H}$ has constant volume and aspect ratio, the number of tetrahedrons within constant distance to the boundary of R is $O(r^{2/3})$. It remains to bound the number of tetrahedrons within constant distance to the boundary of $\mathcal{T}$ that's contained in R.

Define $S \stackrel{\text{def}}{=} \mathcal{T} \cap R$. Since both $\mathcal{T}$ and R are convex, S is convex. Let Surf$(\cdot)$ the surface area of a shape. By Observation 5.4.1, the number of boundary tetrahedrons of $\mathcal{T}$ contained in R is $O(\text{Surf}(S))$.

Let B_1 be the smallest ball containing S. Since R is a cube, we have

$$\text{Surf}(B_1) = O(\text{Surf}(R)).$$

So it suffices to show $\text{Surf}(S) \leq \text{Surf}(B_1)$. We do so by giving a one-to-one mapping of every point from the surface of S onto B_1.

Let $\phi : S \to B_1$ which maps each face of S to a subset of the surface of B_1, defined as follows. Consider a face of S, say f, with vertices $v_1, \ldots, v_k$ in a clockwise order. Let P_f be the plane containing f. P_f cuts B_1 into two parts, let B_1' be the part of the smaller volume (break a tie arbitrarily), aka the sphere cap generated by the plane p_f.

For each $1 \leq i \leq k$, let P_i be the plane orthogonal to P_f that passes through v_i and v_{i+1}) (if $i = k$, then the intersection line is (v_k, v_1)). Let u_i be the point of intersection of the surface of B_1 with the planes P_{i-1} and P_i (if $i = 1$, then u_1 is the intersect vertex of P_1, P_k and the surface of B_1').

We define $\phi(f)$ to be the surface of B_1' enclosed by $(u_1, \ldots, u_k, u_1)$. See Figure 5.4 for an example.

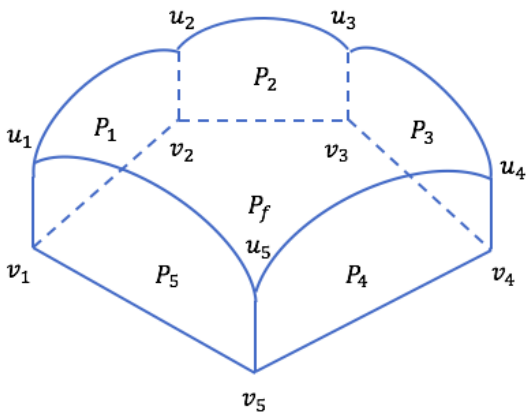

Figure 5.4: An example of $\phi(f)$. Face f has five vertices $v_1, \ldots, v_5$. $\phi(f)$ is the surface of B_1' enclosed by $(u_1, \ldots, u_5, u_1)$.

For each face f of S, the orthogonal projection of $\phi(f)$ onto the plane P_f is f. Thus

$$\textsc{Area}(\phi(f)) \geq \textsc{Area}(f).$$

In addition, since S is convex, for any two distinct faces f_1 and f_2, $\phi(f_1)$ and $\phi(f_2)$ are also disjoint. So we have

$$\text{Surf}\left(S\right) = \sum_{f \in S} \textsc{Area}\left(f\right) \leq \sum_{f \in S} \textsc{Area}\left(\phi\left(f\right)\right) \leq \text{Surf}\left(B_1\right).$$

Combining this with $\text{Surf}(B_1) \leq O(\text{Surf}(S)) \leq O(r^{2/3})$ then completes the proof. □

5.4.2 Bounding the Number of Tetrahedrons in $\mathcal{H}$ Intersecting with a Plane

In this section, we show that $\mathcal{H} = \textsc{Hollow}(\mathcal{T}, B, r)$, computed by Algorithm 13, has a small overlap with any plane whose normal vector has an angle between $(0, \pi/2)$ with the longest direction of B. That is, $\mathcal{H}$ satisfies the second condition of the (B, r)-hollowing definition in Definition 5.1.2.

Lemma 5.4.3. *Given a convex edge-simple 3D truss $\mathcal{T}$ of n vertices, a bounding box B and a positive integer $r \leq n/\alpha(\mathcal{T})^2$, let $\mathcal{H} = \textsc{Hollow}(\mathcal{T}, B, r)$ returned by Algorithm 13. Let $\mathbf{d} \in \mathbb{R}^3$ be a unit vector such that the angle between $\mathbf{d}$ and let the angles with the three directions of the box (normals to its faces) be $\theta_x, \theta_y, \theta_z \in (\theta, \pi/2)$, for some $\theta > 0$. Then, the number of tetrahedrons in $\mathcal{H}$ which intersect any plane P orthogonal to $\mathbf{d}$ is at most*

$$O\left(\frac{n^{2/3}}{\alpha(\mathcal{T})^{1/3} r^{1/3} \cos^2\theta}\right).$$

Without loss of generality, we assume that the bounding box B is *axis-parallel*, that is, the sides of B are parallel to the three axes: the x axis, the y axis and the z-axis. We say an axis-paralleled box has side lengths $a, b, c > 0$, if the sides parallel to the x-axis have

length a, the sides parallel to the y-axis have length b, and the sides parallel to the z-axis have length c, respectively.

To prove Lemma 5.4.3, we need the following claim, which bounds the intersection area of a 2D plane and a 3D box.

Claim 5.4.4. *Let B be a 3D axis-parallel box of side lengths $a, b, c > 0$. Let $\mathbf{d} \in \mathbb{R}^3$ be a unit vector such that the angle between $\mathbf{d}$ and the x-axis (the y-axis, and the z-axis) is θ_x (θ_y, θ_z, respectively). Suppose $\theta_x, \theta_y, \theta_z \in (0, \pi/2)$. Then the intersection area S of any 2D plane P orthogonal to $\mathbf{d}$ and the 3D box B satisfies*

$$S \leq \min\left\{\frac{bc}{\cos\theta_x}, \frac{ac}{\cos\theta_y}, \frac{ab}{\cos\theta_z}\right\}.$$

Proof. Since the three terms in the right hand side are symmetric, we only prove S can be upper bounded by the first term and the other two follow in a similar way.

Let B_{x1}, B_{x2} be the two faces of B which are orthogonal to the x-axis, without loss of generality, assume B_{x1} has a smaller x-coordinate.

If P intersects neither B_{x1} nor B_{x2}, then the volume of B is equal to $Sa\cos\theta_x$.

If P intersects B_{x1} say with line (K, L), see Figure 5.5, then we draw a line going through point K and parallel to the x-axis, which intersects face B_{x2} at point M, similarly we draw a line going through point L and parallel to the x-axis, which intersects face B_{x2} at point N. We cut the box B by the plane $KMLN$, see Figure 5.5. Note that the volume of the convex hull of (K, G, E, F, G, H, I, J), the right one in Figure 5.5, equals to $Sa\cos\theta_x$, which is smaller than the volume of B.

Similarly, if P intersects B_{x2}, then we can draw two lines parallel to the x-axis and going through the two intersection points respectively and get a shape of volume $Sa\cos\theta_x$ smaller than the volume of B. Note that the intersection between P and B_{x2} cannot coincide with (M, N), otherwise the angle $\theta_x = \pi/2$. We can check that the shape we get

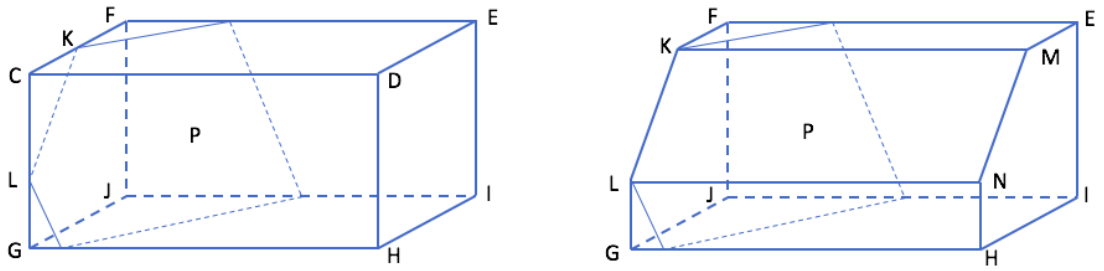

Figure 5.5: Plane P intersects face B_{x1} with line (K, L). We draw a line going through point K and parallel to the x-axis, which intersects face B_{x2} at point M, similarly we draw a line going through point L and parallel to the x-axis, which intersects face B_{x2} at point N.

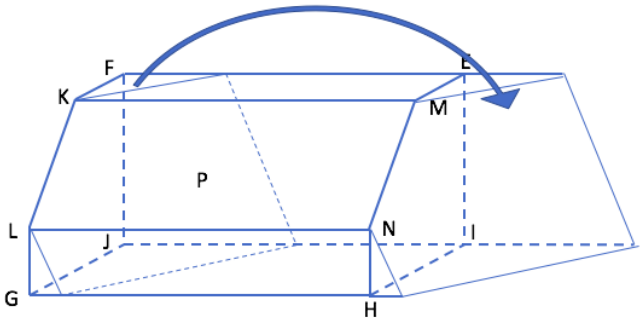

Figure 5.6: Cut the shape along the plane P, and shift the left part along the x-axis and glue the two faces $KFJGL$ and $MEIHN$.

must contain $P \cap B$, and the two faces of the shape which are orthogonal to the x-axis are congruent. If we cut the shape along the plane P, then we can shift the left part along the x-axis and glue the two faces which are orthogonal to the x-axis, and get a parallelepiped which has $P \cap B$ as a face. Figure 5.6 shows the parallelepiped we get from the example of Figure 5.5. Thus,

$$S \cdot a \cdot \cos\theta_x \leq abc.$$

That is, $S \leq bc/\cos\theta_x$. This completes the proof. □

Now we prove Lemma 5.4.3.

Proof of Lemma 5.4.3. Let t be the number of regions (that is, small cubes of the hollowing) of $\mathcal{H}$ which intersect a 2D plane P, and let m be the maximum number of tetrahedrons of a single region of $\mathcal{H}$ which intersect the plane P. The number of tetrahedrons in the hollowing $\mathcal{H}$ which intersect P can be upper bounded by tm.

We first bound t, the number of regions of $\mathcal{H}$ which intersect the plane P. A cube region intersects the plane P only if all its points are within a distance $\sqrt{3}r^{1/3}$ of P. We put two planes, say P_1, P_2, which are parallel to the plane P, above and below P with distance $\sqrt{3}r^{1/3}$ to P. All cube regions which intersect the plane P must be within the two planes P_1 and P_2. By Claim 5.4.4, the volume of the box B between the two planes P_1, P_2 is at most

$$\frac{bc}{\cos\theta} \cdot 2\sqrt{3}r^{1/3}.$$

Since each cube has volume at most r, we can bound the number of cube regions which intersect the plane P

$$t \leq \frac{2\sqrt{3}bc}{r^{2/3}\cos\theta}.$$

We then bound m, the maximum number of tetrahedrons of a single region of $\mathcal{H}$ which intersect the plane P. Consider the tetrahedrons of $\mathcal{H}$ in this single cube region which intersect a some hollowing plane. These tetrahedrons are within constant distance of an $r^{1/3} \times r^{1/3}$ square, which is on a plane orthogonal to one of the direction of the bonding box. By Observation 5.4.1 and Claim 5.4.4, the number of these tetrahedrons which intersect the plane P is at most $O(r^{1/3}/\cos\theta)$. Thus, we can bound the maximum number of tetrahedrons of a single cube region which intersect the plane P

$$m \leq O\left(\frac{r^{1/3}}{\cos\theta}\right).$$

Therefore, the number of tetrahedrons of the hollowing $\mathcal{H}$ which intersect the plane P is at

most

$$tm \leq O\left(\frac{bc}{r^{1/3}\cos^2\theta}\right).$$

Note that we have $a = c \cdot \alpha(\mathcal{T})$ and $a \geq b \geq c$. The above number is upper bounded by

$$O\left(\frac{n^{2/3}}{\alpha(\mathcal{T})^{1/3} r^{1/3}\cos^2\theta}\right).$$

□

5.4.3 Bounding the Relative Condition Number of $\mathcal{H}$

In this section, we show that the condition number of $\boldsymbol{A}_{\mathcal{H}}$ and $\textsc{Sc}[\boldsymbol{A}_{\mathcal{T}}]_U$ is small. That is, $\mathcal{H}$ satisfies the third condition of the (B, r)-hollowing definition in Definition 5.1.2.

Since $\mathcal{T}$ is convex, each region of $\mathcal{H}$ is connected. Lemma 5.1.1 implies that $\boldsymbol{A}_{\mathcal{H}}$ and $\textsc{Sc}[\boldsymbol{A}_{\mathcal{T}}]_U$ has the same null space. By Lemma 5.1.1, the smallest nonzero eigenvalue of $\boldsymbol{A}_{\mathcal{H}}$ is lower bounded by the diameter and the size of each region of $\mathcal{H}$.

Lemma 5.4.5. *Let $\boldsymbol{A}_{\mathcal{T}}$ be the truss stiffness matrix of a convex edge-simple 3D truss $\mathcal{T}$. Given a positive integer r and a bounding box B, let $\mathcal{H} = \textsc{Hollow}(\mathcal{T}, B, r)$ be returned by Algorithm 13. Let $\boldsymbol{A}_{\mathcal{H}}$ be the associated truss stiffness matrix of $\mathcal{H}$. Then,*

$$\boldsymbol{A}_{\mathcal{H}} \preccurlyeq \textsc{Sc}[\boldsymbol{A}_{\mathcal{T}}]_U \preccurlyeq O(r^2)\boldsymbol{A}_{\mathcal{H}},$$

where U consists of all vertices in $\mathcal{H}$.

Proof. The first inequality is equivalent to: $\textsc{Sc}[\boldsymbol{A}_{\mathcal{T}}]_U - \boldsymbol{A}_{\mathcal{H}}$ is a symmetric PSD matrix. Let $\mathcal{T}'$ be the truss obtained by removing all the edges of $\mathcal{T}$ whose two endpoints are both in U. Let $\boldsymbol{A}_{\mathcal{T}'}$ be the associated truss stiffness matrix of $\mathcal{T}'$. We can check that

$$\textsc{Sc}[\boldsymbol{A}_{\mathcal{T}}]_U - \boldsymbol{A}_{\mathcal{H}} = \textsc{Sc}[\boldsymbol{A}_{\mathcal{T}'}]_U.$$

By Fact 2.1.3, $\text{SC}[\boldsymbol{A}_{\mathcal{T}}]_U - \boldsymbol{A}_{\mathcal{H}}$ is a symmetric PSD matrix.

It remains to prove the second inequality. Note the planes of Algorithm 13 line 1 divide $\mathcal{T}$ into small regions. Let $\mathcal{T}_i$ denote the subgraph induced by $\mathcal{T}$ on the ith region, and let $\mathcal{T}_{C_i}$ denote the subgraph induced by $\mathcal{T}$ on the boundary of the ith region. Let $\boldsymbol{A}_{\mathcal{T}_i}, \boldsymbol{A}_{\mathcal{T}_{C_i}}$ be the associated truss stiffness matrices of $\mathcal{T}_i$ and $\mathcal{T}_{C_i}$ respectively. Let U_i denote the boundary vertices of the ith region. Let $\text{SC}[\boldsymbol{A}_{\mathcal{T}_i}]_{U_i}$ be the Schur complement of $\boldsymbol{A}_{\mathcal{T}_i}$ w.r.t. to U_i.

Since $\mathcal{T}$ is a convex edge-simple 3D truss, each vertex in $\mathcal{T}_i$ has constant degree and each edge has constant length and elasticity parameter. It implies $\lambda_{\max}(\boldsymbol{A}_{\mathcal{T}_i}) = O(1)$. By Fact 2.1.3, $\lambda_{\max}(\text{SC}[\boldsymbol{A}_{\mathcal{T}_i}]_{U_i}) = O(1)$.

According to Algorithm 13 from line 2 to line 4, in each $\mathcal{T}_{C_i}$, the tetrahedrons are arranged in simplicial complex and $\mathcal{T}_{C_i}$ is connected. By Lemma 5.1.1, the null spaces of $\boldsymbol{A}_{\mathcal{T}_{C_i}}$ and the null space of $\text{SC}[\boldsymbol{A}_{\mathcal{T}_i}]_{U_i}$ are the same. Besides, each $\mathcal{T}_{C_i}$ has $O(r^{2/3})$ vertices and diameter $O(r^{1/3})$, by Lemma 5.4.2. Applying Lemma 5.1.1 gives:

$$\lambda_{\min}(\boldsymbol{A}_{\mathcal{T}_{C_i}}) = \Omega\left(\frac{1}{r^2}\right).$$

This implies:

$$\text{SC}[\boldsymbol{A}_{\mathcal{T}_i}]_{U_i} \preccurlyeq O(r^2)\boldsymbol{A}_{\mathcal{T}_{C_i}}.$$

Note each edge of $\mathcal{T}$ only appears in a constant number of regions. Thus,

$$\text{SC}[\boldsymbol{A}_{\mathcal{T}}]_U \preccurlyeq \sum_i \text{SC}[\boldsymbol{A}_{\mathcal{T}_i}]_{U_i} \preccurlyeq O(r^2)\sum_i \boldsymbol{A}_{\mathcal{T}_{C_i}} \preccurlyeq O(r^2)\boldsymbol{A}_{\mathcal{H}}.$$

This completes the proof. □

5.5 Nested Dissection

In this section, we present our solver for the constructed preconditioners, Algorithm 15 CONVEXTRUSSUNIONND, which is used as a subroutine in line 8 of Algorithm 12 TRUSS-

SOLVER.

We first prove Lemma 2.5.2 in Section 5.5.1. Then, in Section 5.5.2, we use Lemma 2.5.2 to analyze the performance of Algorithm 15, which proves Lemma 5.1.5.

5.5.1 Proof of Lemma 2.5.2

We restate Lemma 2.5.2 with running time phrased in terms of the matrix multiplication exponent ω.

Lemma 2.5.2. *Suppose we have a recursive separator decomposition of a simplicial complex with n bounded aspect ratio tetrahedrons such that:*

1. *the number of leaves, and hence total number of recursive calls, is at most n^{α}.*
2. *each leaf (bottom layer partition) has at most n^{β} tetrahedrons.*
3. *each top separator has size at most n^{γ}.*

Then we can find an exact Cholesky factorization of the associated stiffness matrix with multiplication count $O(n^{\alpha+2\omega\beta/3} + n^{\alpha+\gamma\omega}\log n)$ and fill-in size $O(n^{\alpha+\frac{4}{3}\beta} + n^{\alpha+2\gamma})$.

Nested dissection according to the separator decomposition stated in Lemma 2.5.2 has three parts of cost:

1. Inverting leaf components:
   (a) the cost only associated with vertices not belonging to top-level separators;
   (b) the cost associated with top-level separators.
2. Inverting top-level separators.

We first analyze the cost of inverting top-level separators. Note that top-level separators are disjoint. After eliminating all leaf components, we get a *layered graph*. That is, its vertices can be partitioned into $V_1 \cup \ldots \cup V_l$ for some positive integer l such that there is

no edge between V_i and V_j for $|i-j|>1$. Algorithm 14 gives a numbering for a layered graph, and Claim 5.5.1 analyzes its performance.

Algorithm 14 LAYEREDGRAPHND($G=(V_1\cup\ldots\cup V_l,E)$)

Input: a layered graph G with l layers
Output: an elimination ordering for vertices in G

1: **if** $l=1$ **then**
2: number the vertices in V_1 arbitrarily and return this numbering.
3: **end if**
4: Label $V_{\lfloor l/2\rfloor}$ with the highest possible numbers.
5: **return** the numbering of LAYEREDGRAPHND($G[V_1\cup\ldots\cup V_{\lfloor l/2\rfloor-1}]$ and LAYEREDGRAPHND($G[V_{\lfloor l/2\rfloor+1}\cup\ldots\cup V_l]$, and $V_{\lfloor l/2\rfloor}$.

Claim 5.5.1. *Let G be a layered graph with l layers of at most s vertices each. Algorithm 14* LAYEREDGRAPHND *returns a numbering such that, Gaussian elimination according to this order has fill-in size $O(s^2l)$ and multiplication count $O(s^\omega l)$.*

Proof. We follow the proofs of [48]. We first bound the fill-in size. Consider a recursion of Algorithm 14 on a graph with n vertices, let $f(n)$ denote the maximum number of fill-in edges whose lower numbered endpoint is numbered by this recursion. Suppose this recursion deals with layers $V_s, V_{s+1}\ldots,V_t$. The algorithm numbers the vertices in the middle layer, that is, $V_{\lfloor(s+t)/2\rfloor}$, which can be viewed as a separator whose removal separates the graph into two disjoint and balanced parts. The fill-in edges whose lower numbered endpoint is in $V_{\lfloor(s+t)/2\rfloor}$ consists of the following edges: (1) edges whose both endpoints are in $V_{\lfloor(s+t)/2\rfloor}$; and (2) edges whose one endpoint is in $V_{\lfloor(s+t)/2\rfloor}$ and the other is in V_{s-1} (if exists) or V_{t+1} (if exists). Since each layer has at most s vertices,

$$f(n)\le\begin{cases}O(s^2), & \text{if } n=O(s)\\ f(n_1)+f(n_2)+3s^2, & \text{otherwise}\end{cases}$$

Note $n_2\le n_1\le n_2+s$. Thus, the total fill-in size is $O(s^2l)$.

We then bound the multiplication count. For a recursion of Algorithm 14 on a graph

with n vertices, let $g(n)$ denote the maximum multiplication count associated with vertices in this graph which are going to be numbered in this recursion. Similar to the analysis of the fill-in, we have

$$g(n) \leq \begin{cases} O(s^\omega), & \text{if } n = O(s) \\ g(n_1) + g(n_2) + 3s^\omega, & \text{otherwise} \end{cases}$$

Thus, the total multiplication count is $O(s^\omega l)$. □

Using Algorithm 14 as a subroutine, we prove Lemma 2.5.2.

Proof of Lemma 2.5.2. We first bound the fill-in size. Note each leaf component has $O(n^\beta)$ vertices in which $O(n^\gamma)$ vertices are in some top-level separators and are labeled by higher numbers. By the result in [48] and [49], the fill-in introduced by inverting each leaf component is

$$O\left(n^{4\beta/3} + n^{2\beta/3+\gamma}\right).$$

There are totally $O(n^\alpha)$ leaf components. Thus, the fill-in size introduced by inverting all leaf components is $O\left(n^{\alpha+4\beta/3} + n^{\alpha+2\beta/3+\gamma}\right)$.

By Claim 5.5.1, the fill-in size of inverting top-level separators is $O(n^{\alpha+2\gamma})$. Thus, the total fill-in size is

$$O\left(\alpha + n^{4\beta/3} + n^{\alpha+2\beta/3+\gamma} + n^{\alpha+2\gamma}\right) = O\left(n^{\alpha+4\beta/3} + n^{\alpha+2\gamma}\right).$$

Here, we use Cauchy-Schwarz inequality to drop the second term.

We then bound the multiplication count. By [49], when inverting each leaf component, the multiplication count only associated with vertices not belonging to top-level separators is $O(n^{2\omega\beta/3})$. By Claim 5.5.1, the multiplication count of inverting top-level separators is $O(n^{\alpha+\omega\gamma})$.

It remains to upper bound the multiplication count associated with top-level separators

when inverting a leaf component. We adapt the analysis in [48].

We prove a more general result. Consider a tetrahedral mesh G with n vertices, in which s vertices are in a top-level separator and have been labeled with higher numbers. Let $g(n, s)$ be the multiplication count associated these s top-level vertices when running nested dissection on G.

Nested dissection finds a separator of size $n^{2/3}$ for G. The multiplication count associated with the s top-level vertices when inverting this separator can be upper bounded by $O((n^{2/3} + s)^\omega)$. This gives the following recursion:

$$\begin{aligned} g(n, s) &\leq O((n^{2/3} + s)^\omega) + \max_{n_i, s_i} \{ \sum_{1 \leq i \leq 2} g(n_i, s_i) \} \\ &\leq O(2^\omega n^{2\omega/3} + 2^\omega s^\omega) + \max_{n_i, s_i} \{ \sum_{1 \leq i \leq 2} g(n_i, s_i) \}. \end{aligned}$$

Here, the maximum is taken over

$$\begin{aligned} s_1 + s_2 &\leq s + n^{2/3} \\ n_1 + n_2 &\leq n + n^{2/3} \\ n_1, n_2 &\leq \frac{4}{5} n + n^{2/3}. \end{aligned}$$

We can compute that

$$g(n, s) = O\left(n^{2\omega/3} + s^\omega \log n\right).$$

Note each leaf component is incident to at most two top-level separators. Thus, the multiplication count associated with the top-level vertices when inverting this leaf component is

$$O\left(n^{2\omega\beta/3} + n^{\gamma\omega} \log n\right).$$

Combining the other two parts of cost, and the fact that there are totally n^α leaf components,

the total multiplication count is

$$O\left(n^{\alpha+2\omega\beta/3} + n^{\alpha+\gamma\omega}\log n\right).$$

If we replace ω by 3, then we can drop the $\log n$ factor in the above equation, which gives the result in the statement. □

5.5.2 Proof of Lemma 5.1.5

We now present Algorithm 15 CONVEXTRUSSUNIONND. The input consists of a edge-simple 3D truss $\mathcal{T}$ which is a union of k convex edge-simple trusses, a bounding box for each convex edge-simple truss, the index subset of small-aspect-ratio trusses and (B_i, r_i)-hollowings for each small-aspect-ratio truss. The output is an elimination ordering for the union of the hollowings of small-aspect-ratio trusses and the large-aspect-ratio trusses. This proves Lemma 5.1.5.

We first prove that there exists a good direction $\boldsymbol{d}$ such that: the angle between $\boldsymbol{d}$ and the longest direction of each bounding box is in a proper range.

Lemma 5.5.2. *Let $k \geq 2$ and $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_k \in \mathbb{R}^3$ be k unit vectors. Then there exists a unit vector $\boldsymbol{d} \in \mathbb{R}^3$ such that*

$$\frac{1}{10k} \leq \left|\boldsymbol{d}^\top \boldsymbol{d}_i\right| \leq 1 - \frac{1}{10k}, \forall 1 \leq i \leq k. \tag{5.17}$$

Proof. We pick a unit vector $\boldsymbol{d}$ uniformly at random. For any fixed i,

$$\Pr\left(\frac{1}{10k} \leq \left|\boldsymbol{d}^\top \boldsymbol{d}_i\right| \leq 1 - \frac{1}{10k}\right) = 2 \cdot \frac{\text{vol}(1/10k) - \text{vol}(1 - 1/10k)}{V}.$$

Here, $\text{vol}(x)$ is the volume of a cap of a 3D unit ball with height $1 - x$, and V is the volume of a 3D unit ball. We have $\text{vol}(x) = \frac{1}{6}\pi(1 - x^2)(3(1 - x^4) + (1 - x^2)^2)$ and $V = \frac{4}{3}\pi$.

Algorithm 15 CONVEXTRUSSUNIONND($\mathcal{T} = \langle \{\boldsymbol{p}_i\}_{i \in V}, T, E, \gamma \rangle, \mathcal{B}, \mathcal{I}, \mathcal{H}, l$)

Input: a edge-simple 3D truss $\mathcal{T}$ which is the union of k convex edge-simple trusses $\mathcal{T}_1, \ldots, \mathcal{T}_k$,
$\mathcal{B} = \{B_1, \ldots, B_k\}$ in which B_i is a bounding box of $\mathcal{T}_i$ constructed from Line 2 of Algorithm 12,
Index set $\mathcal{I}$ of large aspect ratio complexes constructed from Line 4 of Algorithm 12,
Hollowings of each $\mathcal{T}_i$ with $i \in \mathcal{I}$ constructed from Line 6 of Algorithm 12,
l, the number of top-level partitions.

Output: an elimination ordering for vertices in $(\cup_{i \in \mathcal{I}} \mathcal{H}_i) \bigcup (\cup_{i \notin \mathcal{I}} \mathcal{T}_i)$

1: For each $1 \leq i \leq k$, $\boldsymbol{d}_i \leftarrow$ the direction of the longest sides of B_i.
2: Compute a unit vector $\boldsymbol{d}$ such that $\frac{1}{10k} \leq \left| \boldsymbol{d}^\top \boldsymbol{d}_i \right| \leq 1 - \frac{1}{10k}, \forall 1 \leq i \leq k$.
3: Compute separator planes $P_1, \ldots, P_l$, which are planes orthogonal to $\boldsymbol{d}$ and dividing $\mathcal{T}$ into $l+1$ parts $Q_1, \ldots, Q_{l+1}$ of $O(nl^{-1})$ tetrahedrons each.
4: For each $1 \leq j \leq l$, $S_j \leftarrow$ tetrahedrons in $(\cup_{i \in \mathcal{I}} \mathcal{H}_i) \bigcup (\cup_{i \notin \mathcal{I}} \mathcal{T}_i)$ which intersect plane P_j.
5: $Q_1 \leftarrow Q_1 \cup S_1, Q_{l+1} \leftarrow Q_{l+1} \cup S_l$.
6: For each $2 \leq j \leq l, Q_j \leftarrow Q_j \cup S_{j-1} \cup S_j$.
7: LAYEREDGRAPHND($\mathcal{T}[S_1 \cup \ldots \cup S_l]$) with highest numbers.
8: Run nested dissection with MT-separators[1] for each Q_j to number its unnumbered vertices.
9: **return** the elimination ordering of vertices in $(\cup_{i \in \mathcal{I}} \mathcal{H}_i) \bigcup (\cup_{i \notin \mathcal{I}} \mathcal{T}_i)$.

Plugging these volumes into the above equation,

$$\Pr\left(\frac{1}{10k} \leq \left| \boldsymbol{d}^\top \boldsymbol{d}_i \right| \leq 1 - \frac{1}{10k}\right) = \frac{1}{2}\left(\left(1 - \frac{1}{10k}\right)^2 \left(2 + \frac{1}{10k}\right) - \left(\frac{1}{10k}\right)^2 \left(3 - \frac{1}{10k}\right)\right).$$

Take the opposite:

$$\Pr\left(\left| \boldsymbol{d}^\top \boldsymbol{d}_i \right| < \frac{1}{10k} \text{ or } \left| \boldsymbol{d}^\top \boldsymbol{d}_i \right| > 1 - \frac{1}{10k}\right) \leq \frac{3}{20k} - \frac{1}{2000k^3} + \frac{3}{200k^2}.$$

By union bound,

$$\Pr\left(\exists i \; s.t. \; \left| \boldsymbol{d}^\top \boldsymbol{d}_i \right| < \frac{1}{10k} \text{ or } \left| \boldsymbol{d}^\top \boldsymbol{d}_i \right| > 1 - \frac{1}{10k}\right) \leq k\left(\frac{3}{20k} - \frac{1}{2000k^3} + \frac{3}{200k^2}\right) \leq \frac{1}{5}.$$

Thus,

$$\Pr\left(\forall i, \frac{1}{10k} \leq \left| \boldsymbol{d}^\top \boldsymbol{d}_i \right| \leq 1 - \frac{1}{10k}\right) \geq \frac{4}{5}.$$

This implies there exists a $\boldsymbol{d}$ as desired. □

We independently pick $O(\log n)$ unit vectors uniformly at random. By a Chernoff bound, we can find a direction $\boldsymbol{d}$ satisfying Equation (5.17) with high probability.

Recall that Lemma 5.4.3 states that: any 2D plane P orthogonal to $\boldsymbol{d}$ intersects a small number of tetrahedrons in a (B, r)-hollowing of a convex edge-simple 3D truss. Lemma 5.1.4 bounds the number of tetrahedrons in a convex edge-simple 3D truss which intersect a single 2D plane P orthogonal to $\boldsymbol{d}$. We restate it in the following, which can be proved by combining Observation 5.4.1 and Claim 5.4.4.

Lemma 5.1.4. *Let $\mathcal{T}$ be a convex edge-simple 3D truss of n vertices, and let B be a bounding box of $\mathcal{T}$. Let $\boldsymbol{d} \in \mathbb{R}^3$ be a unit vector such that the angle between $\boldsymbol{d}$ and the longest direction of B is $\theta \neq \pi/2$. Then any plane P orthogonal to $\boldsymbol{d}$ intersects $\mathcal{T}$ in at most $O(n^{2/3}\alpha(\mathcal{T})^{-1/3}\cos^{-1}\theta)$ tetrahedrons.*

Lemma 5.4.3 and Lemma 5.1.4 together imply that: the top-level separator $S_1 \cup \ldots \cup S_l$ computed in Algorithm 15 has a small size. This, together with nested dissection in [49], lets us prove that Algorithm 15 outputs an elimination ordering with a small fill-in size and multiplication count.

Lemma 5.5.3. *Given a edge-simple 3D truss $\mathcal{T}$ of n vertices which is a union of k convex edge-simple trusses with n_i vertices each, running Algorithm 12,* TRUSSSOLVER$(\mathcal{T}, \boldsymbol{f}, \epsilon, c_\alpha, c_r)$*, with Line 8 replaced by Algorithm 15,* CONVEXTRUSSUNIONND$(\mathcal{T}, \mathcal{B}, \mathcal{I}, \mathcal{H}, l)$*, leads to performance in terms of n that is optimized by setting*

$$c_\alpha = c_r \leq \frac{1}{3}$$

in Line 4 of Algorithm 12, TRUSSSOLVER*. In terms of c_r, the hollowing parameter, and l, the number of top-level separators, this gives an elimination ordering with fill-in size at most*

$$O(n^{4/3}l^{-1/3} + k^{14/3+2c_r/3}n^{4/3-2c_r/3}l),$$

that can be computed in time

$$O(n^{2\omega/3}l^{-2\omega/3+1} + k^{7\omega/3+\omega c_r/3}n^{2\omega/3-c_r\omega/3}l\log n),$$

where ω is the matrix multiplication exponent.

Proof. We apply Lemma 2.5.2. According to Algorithm 15 line 2, for each i, the angle between the longest direction of the ith bounding box and $\boldsymbol{d}$ has cosine value in $[1/10k, 1-1/10k]$.

We first upper bound the number of vertices in each top-level separators, that is, the number of tetrahedrons in $(\cup_{i\in\mathcal{I}}\mathcal{H}_i)\bigcup(\cup_{i\notin\mathcal{I}}\mathcal{T}_i)$ which intersects a plane P_j, see Algorithm 15 line 4. For each $i \in \mathcal{I} \stackrel{\text{def}}{=} \{i \in [k] : \alpha(\mathcal{T}_i) \leq n_i^{c_\alpha}\}$, by Lemma 5.4.3, the number of tetrahedrons in $\mathcal{H}_i$ intersect a plane P_j is at most

$$O\left(k^2 n_i^{2/3}\alpha_i^{-1/3}r_i^{-1/3}\right) = O\left(k^2 n_i^{2/3-c_r/3}\right),$$

since $\alpha_i \geq 1$. For each $i \notin \mathcal{I}$, by Lemma 5.1.4, the number of tetrahedrons in $\mathcal{T}_i$ intersect a plane P_j is at most

$$O\left(kn_i^{2/3}\alpha_i^{-1/3}\right) = O\left(kn_i^{2/3-c_\alpha/3}\right)$$

since $\alpha_i > n_i^{c_\alpha}$. The two terms have same exponent for n when we set $c_r = c_\alpha$. Note Algorithm 13 requires that $c_r + 2c_\alpha \leq 1$. Thus, here we need $c_r \leq 1/3$.

Thus, the total number of tetrahedrons in $(\cup_{i\in\mathcal{I}}\mathcal{H}_i)\bigcup(\cup_{i\notin\mathcal{I}}\mathcal{T}_i)$ which intersect a single separator plane P_j is then at most:

$$s \stackrel{\text{def}}{=} O\left(\sum_{1\leq i\leq k} k^2 n_i^{2/3-c_r/3}\right) = O\left(k^{7/3+c_r/3}n^{2/3-c_r/3}\right).$$

The last inequality is by Jensen's inequality.

There are totally l top-level separators, which separates the whole truss into $l+1$ sep-

arate components and each component has $O(n/l)$ vertices, according to Algorithm 15 line 3.

We plug these parameters into Lemma 2.5.2, the total fill-in size is

$$O\left(\left(\frac{n}{l}\right)^{4/3} l + s^2 l\right) = O(n^{4/3} l^{-1/3} + k^{14/3+2c_r/3} n^{4/3-2c_r/3} l),$$

and the multiplication count is

$$O\left(\left(\frac{n}{l}\right)^{2\omega/3} l + s^{\omega} l \log n\right) = O(n^{2\omega/3} l^{-2\omega/3+1} + k^{7\omega/3+\omega c_r/3} n^{2\omega/3-c_r\omega/3} l \log n).$$

This completes the proof. □

5.6 Running Times In Terms of Fast Matrix Multiplication

We now restate the running times of our algorithms in terms of faster matrix multiplication / inversion routines. Specifically, we assume inverting an $n \times n$ matrix takes time $O(n^{\omega})$, where $\omega < 2.3728639$ [17].

We first examine purely nested dissection based algorithms. The running time of these algorithms are dominated by the cost of inverting the matrix at the top-most level. Thus, the running time of 3-D nested dissection from [49] as given in Theorem 2.5.1 is

$$O\left(n^{2\omega/3}\right),$$

while the performance of Lemma 2.5.2 becomes

$$O\left(n^{2\omega\beta/3+\alpha} + n^{\omega\gamma+\alpha}\right).$$

We now propagate these different costs for constructing the nested dissection partial states into our running time analyses.

For the bounded aspect ratio case described in Theorem 1.1.4, recall that the input truss is a union of k convex pieces of n_i vertices each. Hollowing with parameter

$$r_i = n_i^{c_r}$$

now takes time

$$O\left(\sum_i \frac{n_i}{r_i} \cdot r_i^{2\omega/3}\right) = O\left(n^{1+c_r(2\omega/3-1)}\right),$$

while still giving a Schur complement of size

$$O\left(n^{1+c_r/3}\right)$$

on the boundaries, and total boundary size of

$$O\left(k^{c_r/3} n^{1-c_r/3}\right).$$

Theorem 2.5.1 then gives that solving this problem on just the boundary elements takes time

$$O\left(k^{2c_r\omega/9} n^{2(1-c_r/3)\omega/3}\right) = O\left(k^{2c_r\omega/9} n^{2\omega/3-2c_r\omega/9}\right),$$

and results in a total fill-in of

$$O\left(k^{4c_r/9} n^{4/3-4c_r/9}\right).$$

Putting these parameters back into the condition number bound of $O(n^{2c_r})$ gives an iteration count of $O(n^{c_r}\log(1/\epsilon))$, which in turn gives a total cost of

$$\begin{aligned}
&O\left(n^{1+c_r(2\omega/3-1)}\right) + O\left(k^{2c_r\omega/9} n^{2\omega/3-2c_r\omega/9}\right) + O\left(n^{c_r}\log(1/\epsilon)\right) \cdot \left[O\left(n^{1+c_r/3}\right) + O\left(k^{c_r/3} n^{4/3-4c_r/9}\right)\right] \\
&= O\left(n^{1+c_r(2\omega/3-1)} + k^{2c_r\omega/9} n^{2\omega/3-2c_r\omega/9} + n^{1+4c_r/3}\log(1/\epsilon) + k^{c_r/3} n^{4/3+5c_r/9}\log(1/\epsilon)\right).
\end{aligned}$$

We can (slightly) simplify this using the fact that $\omega \leq 3$ to drop the first term: it is always

upper bounded by the third. Also, since $k \leq n$, we will focus on optimizing the exponent on n, that is, we want to pick c_r to minimize the maximum of

$$2\omega/3 - 2c_r\omega/9$$
$$1 + 4c_r/3$$
$$4/3 + 5c_r/9$$

By running an LP solver, we get:

- when $\omega = 3$ this is optimized at $c_r = 1/2$, which gives a total cost of $O(k^{1/3}n^{5/3}\log(1/\epsilon))$.
- when $\omega = 2.3728639$, this is optimized at $c_r = 0.2295553$. Here the exponents on n the three terms are 1.4608640, 1.3060737 and 1.4608640 respectively, and we have $2\omega/9 > 1/3$, so so the total cost is bounded by $O(k^{0.1210452}n^{1.4608641}\log(1/\epsilon))$.

For the more general case from Theorem 1.1.4, combining the bounds from Lemma 5.5.3 with the

- $O(n^{1+c_r(2\omega/3-1)})$ cost of computing the Schur complement of eliminating the innards of the hollowings, and
- the $O(n^{1+c_r/3})$ size of the these Schur complements, and
- $O(n^{c_r}\log(1/\epsilon))$ iteration count of PCG

gives a total cost of:[2]

$$O\left(n^{1+c_r(2\omega/3-1)} + n^{2\omega/3}l^{-2\omega/3+1} + k^{7\omega/3+c_r\omega/3}n^{2\omega/3-c_r\omega/3}l \right.$$
$$\left. + n^{c_r}\log(1/\epsilon)\left(n^{1+c_r/3} + n^{4/3}l^{-1/3} + k^{14/3+2c_r/3}n^{4/3-2c_r/3}l\right)\right).$$

[2]We drop the $\log n$ factor here, given $\log n \ll n^c$ for any constant $c > 0$.

Since $\omega \leq 3$, we drop the first term. We can simplify this by moving k and $\log(1/\epsilon)$ to the outermost, and only optimizing the remaining terms:

$$O\left(k^{7\omega/3+c_r\omega/3}\log\left(1/\epsilon\right)\right)\cdot\left(n^{2\omega/3}l^{-2\omega/3+1}+n^{2\omega/3-c_r\omega/3}l+\left(n^{1+4c_r/3}+n^{4/3+c_r}l^{-1/3}+n^{4/3+c_r/3}l\right)\right).$$

Let $l = n^{c_l}$. Since Algorithm 13 requires that $c_r + 2c_\alpha \leq 1$ and in Lemma 5.5.3 we set $c_r = c_\alpha$, we have $0 \leq c_r \leq 1/3$. Subject to this requirement, we minimize the maximum of the following terms:

$$\begin{aligned}
&1+\left(\frac{2\omega}{3}-1\right)c_r\\
&\frac{2\omega}{3}+\left(-\frac{2\omega}{3}+1\right)c_l,\\
&\frac{2\omega}{3}-\frac{\omega}{3}c_r+c_l,\\
&1+\frac{4}{3}c_r,\\
&\frac{4}{3}+c_r-\frac{1}{3}c_l,\\
&\frac{4}{3}+\frac{1}{3}c_r+c_l.
\end{aligned}$$

By running an LP solver, we get:

- when $\omega = 3$ this is optimized at $c_r = 1/3$ and $c_l = 1/6$, which gives a total cost of $O(k^{22/3}n^{11/6}\log(1/\epsilon))$ ($11/6 \approx 1.8333$)
- when $\omega = 2.3728639$, an optimum solution is $c_r = 0.2210963$ and $c_l = 0.1105482$ for a total cost of $O(k^{5.7115596}n^{1.5175803}\log(1/\epsilon))$.

CHAPTER 6

PARALLEL ALGORITHM FOR MIXED PACKING AND COVERING LPS

In this chapter, we prove Theorem 1.1.6.

6.1 Converting an Optimization Problem to Feasibility Problems

Recall a mixed packing and covering LP has the form:

$$\min_{\boldsymbol{x}\geq \mathbf{0}}\{\lambda : \boldsymbol{P}\boldsymbol{x} \leq \lambda \boldsymbol{p}, \boldsymbol{C}\boldsymbol{x} \geq \boldsymbol{c}\}.$$

By a standard binary search, one can reduce $(1+\epsilon)$-approximately solving the above optimization problem into approximately solving $\widetilde{O}(1)$ feasibility problems (e.g., see [13]). Specifically, the feasibility problem is the following: either find $\boldsymbol{x} \geq 0$ such that

$$0 < \max \boldsymbol{P}\boldsymbol{x} \leq (1+\epsilon)\min \boldsymbol{C}\boldsymbol{x} \tag{6.1}$$

or conclude the following LP is infeasible

$$\begin{aligned} \boldsymbol{C}\boldsymbol{x} &\geq \mathbf{1} \\ \boldsymbol{P}\boldsymbol{x} &\leq (1-10\epsilon)\mathbf{1} \\ \boldsymbol{x} &\geq 0 \end{aligned} \tag{6.2}$$

To certify that (6.2) is infeasible, we rely on its dual LP.

Lemma 6.1.1. *By duality,* (6.2) *is infeasible if there exists* $\boldsymbol{y}, \boldsymbol{z} \geq 0$ *s.t.*

$$(1-10\epsilon)\frac{\boldsymbol{C}^\top \boldsymbol{z}}{\mathbf{1}^\top \boldsymbol{z}} < \frac{\boldsymbol{P}^\top \boldsymbol{y}}{\mathbf{1}^\top \boldsymbol{y}}. \tag{6.3}$$

Proof. Eqn. (6.3) is a direct reformulation of the dual LP of (6.2). Since we only need the sufficient condition, the result is by weak duality. If there exists any $\boldsymbol{x} \geq 0$ satisfying (6.2), we have

$$\begin{aligned}(1-10\epsilon)\frac{\boldsymbol{x}^\top \boldsymbol{C}^\top \boldsymbol{z}}{\mathbf{1}^\top \boldsymbol{z}} \geq & (1-10\epsilon)\frac{\mathbf{1}^\top \boldsymbol{z}}{\mathbf{1}^\top \boldsymbol{z}} = 1-10\epsilon, \\ \frac{\boldsymbol{x}^\top \boldsymbol{P}^\top \boldsymbol{y}}{\mathbf{1}^\top \boldsymbol{y}} \leq & (1-10\epsilon)\frac{\mathbf{1}^\top \boldsymbol{y}}{\mathbf{1}^\top \boldsymbol{y}} = 1-10\epsilon.\end{aligned}$$

Together they give $1 < 1$, contradiction. □

6.2 Algorithm for the Feasibility Problem

Our algorithm follows the Lagrangian-relaxation approach as in [13, 78]. The idea is to convert all hard constraints into a single scalar-valued continuous and smoothed function. At each point, we can use a linear function to approximate this smoothed function in a small region.

6.2.1 Potential Function

As in [13], we use the soft-max $\operatorname{lmax}(\boldsymbol{P}\boldsymbol{x})$ and soft-min $\operatorname{lmin}(\boldsymbol{C}\boldsymbol{x})$ to write all constraints into one term. Recall that

$$\operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}) \stackrel{\text{def}}{=} \ln\left(\sum_j \exp(\boldsymbol{P}\boldsymbol{x})_j\right) = \ln\left(\sum_j \exp(\boldsymbol{P}_j^\top \boldsymbol{x})\right),$$

and

$$\operatorname{lmin}(\boldsymbol{C}\boldsymbol{x}) \stackrel{\text{def}}{=} -\ln\left(\sum_j \exp(-\boldsymbol{C}\boldsymbol{x})_j\right) = -\ln\left(\sum_j \exp(-\boldsymbol{C}_j^\top \boldsymbol{x})\right),$$

where $\boldsymbol{P}_j^\top, \boldsymbol{C}_j^\top$ are the j-th row of $\boldsymbol{P}, \boldsymbol{C}$ respectively. In particular, these functions give smooth approximation to $\max \boldsymbol{P}\boldsymbol{x}$ and $\min \boldsymbol{C}\boldsymbol{x}$:

$$\begin{aligned} \max \boldsymbol{P}\boldsymbol{x} \leq \operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}) \leq \max \boldsymbol{P}\boldsymbol{x} + \ln n \\ \min \boldsymbol{C}\boldsymbol{x} \geq \operatorname{lmin}(\boldsymbol{C}\boldsymbol{x}) \geq \min \boldsymbol{C}\boldsymbol{x} - \ln n. \end{aligned} \tag{6.4}$$

When $\operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}) \approx \operatorname{lmin}(\boldsymbol{C}\boldsymbol{x})$, we will have

$$\min \boldsymbol{C}\boldsymbol{x} \leq \max \boldsymbol{P}\boldsymbol{x} \leq \min \boldsymbol{C}\boldsymbol{x} + 2\ln n.$$

If we can control that $\min \boldsymbol{C}\boldsymbol{x} = \Theta(\ln n/\epsilon)$, then the above inequality gives $\max \boldsymbol{P}\boldsymbol{x} \leq (1+\epsilon)\min \boldsymbol{C}\boldsymbol{x}$.

6.2.2 Algorithm

We present our parallel $\tilde{O}(1/\epsilon^3)$ routine in Algorithm 16 for solving the $(1 + O(\epsilon))$-feasibility problem above, that is, either find $\boldsymbol{x} \geq 0$ satisfying (6.1), or certify the infeasibility of (6.2). The input contains a packing constraint matrix $\boldsymbol{P} \in \mathbb{R}_{\geq 0}^{n_P \times m}$, a covering constraint matrix $\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n_C \times m}$, and an error parameter $0 < \epsilon$. That is, there are m variables, n_P packing constraints, and n_C covering constraints. We also use $n = n_P + n_C$ to denote the total number of constraints.

Algorithm 16 starts with small $\boldsymbol{x}_i^{(0)} = \frac{1}{m\|\boldsymbol{P}_{:,i}\|_\infty}, \forall\, i \in [m]$, and keeps increasing $\boldsymbol{x}$ properly, until it reaches the terminate condition in line 4, that is, $\max\{\max \boldsymbol{P}\boldsymbol{x}, \min \boldsymbol{C}\boldsymbol{x}\} \geq K = \frac{10\ln n}{\epsilon}$. The reason of the chosen K value is stated in Lemma 6.3.4. Roughly, when the difference of $\min \boldsymbol{C}\boldsymbol{x}$ and $\max \boldsymbol{P}\boldsymbol{x}$ becomes ϵ factor smaller than $\max\{\max \boldsymbol{P}\boldsymbol{x}, \min \boldsymbol{C}\boldsymbol{x}\}$, we know that $\max \boldsymbol{P}\boldsymbol{x} \leq (1 + O(\epsilon))\min \boldsymbol{C}\boldsymbol{x}$.

In each iteration of the while-loop, we first delete all covering constraints which has already reached K. Since $\boldsymbol{x}$ never decreases, we know that once a row is deleted, we no longer need to look at it. Note the covering matrix cannot be empty, since we enter the

iteration with $\min \boldsymbol{C}\boldsymbol{x} < K$. We compute the vectors $\boldsymbol{y}, \boldsymbol{z}$, which are exponentials of the values of the packing and covering constraints respectively. We then compute $\boldsymbol{a}$ and $\boldsymbol{b}$, which can be considered as gradients of $\mathrm{lmax}(\boldsymbol{P}\boldsymbol{x})$ and $\mathrm{lmin}(\boldsymbol{C}\boldsymbol{x})$ respectively, and use them to guide our update on $\boldsymbol{x}$. In particular, we update $\boldsymbol{x}_i$ if $\boldsymbol{a}_i \leq (1 - \epsilon/50)\boldsymbol{b}_i$ (i.e. $i \in B$). Furthermore, we update $\boldsymbol{x}_i$ multiplicatively by a factor depends on the ratio of $\frac{\boldsymbol{a}_i}{\boldsymbol{b}_i}$, as specified in Eqn. (6.5) and line 11. Note that the smallest update in our algorithm is by a factor of $(1 + \Omega(\frac{\epsilon^2}{\ln n}))$, which is the same as the fixed update step size in [78], and in general our updates take larger steps.

Note that in our analysis, we equivalently view $\boldsymbol{z}$ as the full n_C-dimensional vector, where the coordinates corresponding to deleted constraints are filled by 0's. In particular, the matrix-vector product of the original $\boldsymbol{C}$ with the n_C-dimensional $\boldsymbol{z}$ will be the same as the product of the reduced covering matrix $\boldsymbol{C}^{(t)}$ and reduced $\boldsymbol{z}$.

6.3 Analysis

6.3.1 Proof of Correctness

In this section we will show Algorithm 16 will terminate, and output the correct answer.

Lemma 6.3.1 shows that empty B certifies the infeasibility of the input instance (6.2), which proves the correctness if we end up in the case of line 9.

Lemma 6.3.1. *If the problem instance* (6.2) *is feasible, then*

$$\forall\, \boldsymbol{x} \geq 0, B = \{i : \boldsymbol{a}_i \leq (1 - \frac{\epsilon}{50})\boldsymbol{b}_i\} \neq \emptyset.$$

Proof. Let $\boldsymbol{x}^*$ be a feasible solution of (6.2). Assume by contradiction, $\exists\, \boldsymbol{x} \geq 0, \forall\, i \in [m], \boldsymbol{a}_i^{(t)} > (1 - \frac{\epsilon}{50})\boldsymbol{b}_i^{(t)}$. By definition of $\boldsymbol{a}, \boldsymbol{b}$, it is equivalent to $\exists\, \boldsymbol{y}, \boldsymbol{z} \geq 0$ such that

$$(1 - \frac{\epsilon}{50})\frac{\boldsymbol{C}^\top \boldsymbol{z}}{\boldsymbol{1}^\top \boldsymbol{z}} < \frac{\boldsymbol{P}^\top \boldsymbol{y}}{\boldsymbol{1}^\top \boldsymbol{y}}.$$

Algorithm 16 Parallel algorithm for mixed packing and covering LPs

Input: $\boldsymbol{P}, \boldsymbol{C}, \epsilon$

Output: "infeasible" or $\boldsymbol{x} \geq 0$ s.t. $\boldsymbol{P}\boldsymbol{x} \leq (1+\epsilon)\boldsymbol{C}\boldsymbol{x}$

1: Let $K = \frac{10 \ln n}{\epsilon}$, $\alpha = \frac{1}{K}$, where n is the number of constraints.
2: Initialize $\boldsymbol{x}_i^{(0)} = \frac{1}{m\|\boldsymbol{P}_{:,i}\|_\infty}, \forall\, i \in [m]$, where m is the number of variables.
3: Let $t = 0$.
4: **while** $\max \boldsymbol{P}\boldsymbol{x} < K$ and $\min \boldsymbol{C}\boldsymbol{x} < K$ **do**
5: Let $\boldsymbol{C}^{(t)}$ be $\boldsymbol{C}$ with rows j such that $\left(\boldsymbol{C}\boldsymbol{x}^{(t)}\right)_j \geq K$ deleted.
6: Let $\boldsymbol{y}^{(t)} = \exp\left(\boldsymbol{P}\boldsymbol{x}^{(t)}\right), \boldsymbol{z}^{(t)} = \exp\left(-\boldsymbol{C}^{(t)}\boldsymbol{x}^{(t)}\right)$.
7: $\boldsymbol{a}^{(t)} = \frac{\boldsymbol{P}^\top \boldsymbol{y}^{(t)}}{\mathbf{1}^\top \boldsymbol{y}^{(t)}}, \boldsymbol{b}^{(t)} = \frac{(\boldsymbol{C}^{(t)})^\top \boldsymbol{z}^{(t)}}{\mathbf{1}^\top \boldsymbol{z}^{(t)}}$.
8: Define $B^{(t)} = \{i : \boldsymbol{a}_i^{(t)} \leq (1 - \frac{\epsilon}{50})\boldsymbol{b}_i^{(t)}\}$.
9: If $B^{(t)} = \emptyset$, then return "infeasible".
10: Let

$$\Delta_i^{(t)} = \begin{cases} \frac{1}{2}(1 - \frac{\boldsymbol{a}_i^{(t)}}{\boldsymbol{b}_i^{(t)}}) \in [\epsilon/100, \frac{1}{2}] & \text{if } i \in B^{(t)} \\ 0 & \text{if } i \notin B^{(t)} \end{cases} \tag{6.5}$$

11: $\boldsymbol{x}_i^{(t+1)} \leftarrow \boldsymbol{x}_i^{(t)}(1 + \alpha\Delta_i^{(t)})$.
12: $t \leftarrow t + 1$.
13: **end while**
14: **return** $\boldsymbol{x} = \frac{\boldsymbol{x}^{(t)}}{K}$.

Then the result follows directly from Lemma 6.1.1. □

If Algorithm 16 doesn't terminate with line 9, it must increase at least one variable by at least a factor of $(1 + \frac{\epsilon^2}{10 \ln n})$ each iteration, so the algorithm must reach the termination condition of the while loop at some point, and we need to show the output $\boldsymbol{x}$ satisfies (6.1). We consider the following potential function,

$$f(\boldsymbol{x}^{(t)}) = \operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}^{(t)}) - \operatorname{lmin}(\boldsymbol{C}^{(t)}\boldsymbol{x}^{(t)}) = \ln(\mathbf{1}^\top \boldsymbol{y}^{(t)}) + \ln(\mathbf{1}^\top \boldsymbol{z}^{(t)}).$$

We first quantify the changes of lmax and lmin when we update the variables. This type of smoothness analysis is standard in analyzing algorithms that make updates using gradient information. Similar results are derived in other works on packing and covering (See [54, 13]). The particular analysis we develop can deal with larger gradient steps. In particular,

the approach of our analysis allows updates that may move the gradients of some variables out of their respective coordinate-wise smooth regions, as long as we can still bound the combined impact on the potential function from updates of all variables. This approach can extend straightforwardly to show larger updates also work in [54], and improve their pure packing algorithm to run in $\tilde{O}(\frac{N}{\epsilon^2})$ iterations.

Lemma 6.3.2. *At each iteration t,*

$$\mathrm{lmax}(\boldsymbol{P}\boldsymbol{x}^{(t+1)}) \leq \mathrm{lmax}(\boldsymbol{P}\boldsymbol{x}^{(t)}) + \alpha\langle \boldsymbol{a}^{(t)}, (1+\Delta^{(t)}) \cdot \Delta^{(t)} \cdot \boldsymbol{x}^{(t)} \rangle$$

and

$$\mathrm{lmin}(\boldsymbol{C}^{(t+1)}\boldsymbol{x}^{(t+1)}) \geq \mathrm{lmin}(\boldsymbol{C}^{(t)}\boldsymbol{x}^{(t)}) + \alpha\langle \boldsymbol{b}^{(t)}, (1-\Delta^{(t)}) \cdot \Delta^{(t)} \cdot \boldsymbol{x}^{(t)} \rangle,$$

where $\Delta \cdot \boldsymbol{x}$ is the entry-wise product vector, i.e., $(\Delta \cdot \boldsymbol{x})_i = \Delta_i \boldsymbol{x}_i$.

Proof. To simplify, we omit superscript (t) in the proof.

$$\begin{aligned}
\mathrm{lmax}(\boldsymbol{P}\boldsymbol{x}^{(t+1)}) &= \ln \sum_j \exp(\boldsymbol{P}_j^\top(\boldsymbol{x} + \alpha\Delta \cdot \boldsymbol{x})) \\
&= \ln \sum_j \exp(\boldsymbol{P}_j^\top \boldsymbol{x}) \cdot \exp(\alpha \boldsymbol{P}_j^\top(\Delta \cdot \boldsymbol{x})) \\
&\leq \ln \sum_j \exp(\boldsymbol{P}_j^\top \boldsymbol{x})(1 + \alpha \boldsymbol{P}_j^\top(\Delta \cdot \boldsymbol{x}) + \alpha^2(\boldsymbol{P}_j^\top(\Delta \cdot \boldsymbol{x}))^2).
\end{aligned}$$

Recall $\boldsymbol{P}_j^\top$ is the j-th row of $\boldsymbol{P}$ (i.e. the j-th packing constraint). The last inequality is by Taylor expansion, with $\epsilon \leq \Delta_i \leq \frac{1}{2}$, $\alpha = \epsilon/10\log n$, and $\boldsymbol{P}_j^\top \boldsymbol{x} \leq 10\log n/\epsilon$ for all j, so $\alpha \boldsymbol{P}_j^\top(\Delta \cdot \boldsymbol{x}) \leq \frac{1}{2}$.

We can control the second order term as follows

$$\begin{aligned}
\alpha^2(\boldsymbol{P}_j^\top(\Delta\cdot\boldsymbol{x}))^2 &= \alpha^2\left(\sum_i \boldsymbol{P}_{ji}(\Delta_i\boldsymbol{x}_i)\right)^2 \\
&\le \alpha^2\left(\sum_i \Delta_i\boldsymbol{P}_{ji}(\Delta_i\boldsymbol{x}_i)\right)\left(\sum_i \boldsymbol{P}_{ji}\boldsymbol{x}_i\right) \\
&\le \alpha^2\left(\sum_i \Delta_i\boldsymbol{P}_{ji}(\Delta_i\boldsymbol{x}_i)\right)\frac{10\ln n}{\epsilon} \\
&= \alpha\left(\sum_i \Delta_i\boldsymbol{P}_{ji}(\Delta_i\boldsymbol{x}_i)\right) = \alpha\boldsymbol{P}_j^\top(\Delta\cdot\Delta\cdot\boldsymbol{x}).
\end{aligned}$$

The first inequality is by the Cauchy-Schwarz inequality $\langle u,v\rangle^2 \le \|u\|^2\|v\|^2$, with $u_i = \sqrt{\Delta_i P_{ji}(\Delta_i x_i)}$ and $v_i = \sqrt{P_{ji}x_i}$. The second inequality is due to $\boldsymbol{P}_j^\top\boldsymbol{x} \le \frac{10\ln n}{\epsilon}$ for all j. The last line is by our choice of $\alpha = \frac{\epsilon}{10\ln n}$.

So far we have bounded the impact of the updates on each individual constraint, and we have

$$\mathrm{lmax}(\boldsymbol{P}\boldsymbol{x}^{(t+1)}) \le \ln\sum_j \exp(\boldsymbol{P}_j^\top\boldsymbol{x})(1+\alpha\boldsymbol{P}_j^\top(\Delta\cdot\boldsymbol{x})+\alpha\boldsymbol{P}_j^\top(\Delta\cdot\Delta\cdot\boldsymbol{x})).$$

We then translate the changes on each constraint to the combined change on $\mathrm{lmax}(\boldsymbol{P}\boldsymbol{x})$. Intuitively, the combined change is a convex combination on the changes of each constraint, weighted by their exponential values $\boldsymbol{y}_j = \exp(\boldsymbol{P}_j^\top\boldsymbol{x})$.

$$\begin{aligned}
&\ln\sum_j \exp(\boldsymbol{P}_j^\top\boldsymbol{x})(1+\alpha\boldsymbol{P}_j^\top(\Delta\cdot\boldsymbol{x})+\alpha\boldsymbol{P}_j^\top(\Delta\cdot\Delta\cdot\boldsymbol{x})) \\
=&\ln\sum_j \boldsymbol{y}_j(1+\alpha\boldsymbol{P}_j^\top(\Delta\cdot\boldsymbol{x})+\alpha\boldsymbol{P}_j^\top(\Delta\cdot\Delta\cdot\boldsymbol{x})) \\
=&\ln\left((\mathbf{1}^\top\boldsymbol{y})\left(1+\sum_j\frac{\boldsymbol{y}_j}{\mathbf{1}^\top\boldsymbol{y}}\left(\alpha\boldsymbol{P}_j^\top(\Delta\cdot\boldsymbol{x})+\alpha\boldsymbol{P}_j^\top(\Delta\cdot\Delta\cdot\boldsymbol{x})\right)\right)\right) \\
=&\ln\left((\mathbf{1}^\top\boldsymbol{y})\left(1+\alpha\langle\frac{\boldsymbol{y}}{\mathbf{1}^\top\boldsymbol{y}},\boldsymbol{P}((\mathbf{1}+\Delta)\cdot\Delta\cdot\boldsymbol{x})\rangle\right)\right).
\end{aligned}$$

We can then write out the change of $\mathrm{lmax}(\boldsymbol{P}\boldsymbol{x})$ explicitly as

$$\begin{aligned}
\mathrm{lmax}(\boldsymbol{P}\boldsymbol{x}^{(t+1)}) &\leq \ln\left((\mathbf{1}^\top \boldsymbol{y})\left(1+\alpha\langle\frac{\boldsymbol{y}}{\mathbf{1}^\top\boldsymbol{y}}, \boldsymbol{P}((\mathbf{1}+\Delta)\cdot\Delta\cdot\boldsymbol{x})\rangle\right)\right) \\
&= \ln\left((\mathbf{1}^\top \boldsymbol{y})\left(1+\alpha\langle\frac{\boldsymbol{P}^\top\boldsymbol{y}}{\mathbf{1}^\top\boldsymbol{y}}, (\mathbf{1}+\Delta)\cdot\Delta\cdot\boldsymbol{x}\rangle\right)\right) \\
&= \ln\left((\mathbf{1}^\top \boldsymbol{y})\left(1+\alpha\langle\boldsymbol{a}, (\mathbf{1}+\Delta)\cdot\Delta\cdot\boldsymbol{x}\rangle\right)\right) \\
&= \mathrm{lmax}(\boldsymbol{P}\boldsymbol{x}) + \ln\left(1+\alpha\langle\boldsymbol{a}, (\mathbf{1}+\Delta)\cdot\Delta\cdot\boldsymbol{x}\rangle\right) \\
&\leq \mathrm{lmax}(\boldsymbol{P}\boldsymbol{x}) + \alpha\langle\boldsymbol{a}, (1+\Delta)\cdot\Delta\cdot\boldsymbol{x}\rangle.
\end{aligned}$$

Recall $\boldsymbol{a} = \frac{\boldsymbol{P}^\top\boldsymbol{y}}{\mathbf{1}^\top\boldsymbol{y}}$ as defined in (7), and the last line is by $\ln(1+x) \leq x$ for $x \geq 0$.

For the $\mathrm{lmin}(\boldsymbol{C}^{(t)}\boldsymbol{x})$ part, we follow the same approach.

$$-\mathrm{lmin}(\boldsymbol{C}^{(t+1)}\boldsymbol{x}^{(t+1)}) = \ln\sum_k \exp(-\boldsymbol{C}^{(t+1)}\boldsymbol{x}^{(t+1)})_k,$$

since $\boldsymbol{C}^{(t+1)}$ can only have same or more rows dropped from $\boldsymbol{C}^{(t)}$ due to the overly satisfied constraints, we know $\ln\sum_k \exp(-\boldsymbol{C}^{(t+1)}\boldsymbol{x}^{(t+1)})_k \leq \ln\sum_k \exp(-\boldsymbol{C}^{(t)}\boldsymbol{x}^{(t+1)})_k$. Again omitting the superscript (t), we have

$$\begin{aligned}
-\mathrm{lmin}(\boldsymbol{C}^{(t+1)}\boldsymbol{x}^{(t+1)}) &\leq \ln\sum_k \exp(-\boldsymbol{C}(\boldsymbol{x}+\alpha\Delta\cdot\boldsymbol{x}))_k \\
&= \ln\sum_k \exp(-\boldsymbol{C}\boldsymbol{x})_k \exp(-\alpha\boldsymbol{C}(\Delta\cdot\boldsymbol{x}))_k \\
&\leq \ln\sum_k \exp(-\boldsymbol{C}\boldsymbol{x})_k (1-\alpha\boldsymbol{C}(\Delta\cdot\boldsymbol{x}) + \alpha^2(\boldsymbol{C}(\Delta\cdot\boldsymbol{x}))^2)_k.
\end{aligned}$$

The last inequality is due to $(\alpha\boldsymbol{C}(\Delta\cdot\boldsymbol{x}))_k \leq \frac{1}{2}$ for all k, since in $\boldsymbol{C}^{(t)}$ we only keep those constraints that are not above $\frac{10\ln n}{\epsilon}$ yet, i.e. $(\boldsymbol{C}\boldsymbol{x})_k \leq \frac{1}{\alpha}$ for all k. Again using Cauchy-Schwarz inequality as before and derivations similar to the $\mathrm{lmax}(\boldsymbol{P}\boldsymbol{x})$ case, we get

$$-\mathrm{lmin}(\boldsymbol{C}^{(t+1)}\boldsymbol{x}^{(t+1)}) \leq -\mathrm{lmin}(\boldsymbol{C}^{(t)}\boldsymbol{x}^{(t)}) - \alpha\langle\boldsymbol{b}, (1-\Delta)\cdot\Delta\cdot\boldsymbol{x}\rangle.$$

□

With the above bounds on the changes of the two components $\operatorname{lmax}(\boldsymbol{P}\boldsymbol{x})$ and $\operatorname{lmin}(\boldsymbol{C}\boldsymbol{x})$, we can show how our updates move the potential function $f(\boldsymbol{x})$.

Lemma 6.3.3. *Given* $\max \boldsymbol{P}\boldsymbol{x}^{(t)} < \frac{10 \ln n}{\epsilon}$ *and* $\min \boldsymbol{C}\boldsymbol{x}^{(t)} < \frac{10 \ln n}{\epsilon}$*, we always have* $f(\boldsymbol{x}^{(t)}) \leq 2 \ln n$ *during the execution of Algorithm 16.*

Proof. Initially, $\boldsymbol{x}_i^{(0)} = \frac{1}{m\|\boldsymbol{P}_{:,i}\|_\infty}$, we have $\boldsymbol{P}\boldsymbol{x}^{(0)} \leq \mathbf{1}$ and $\boldsymbol{C}\boldsymbol{x} \geq 0$, thus $f(\boldsymbol{x}^{(0)}) \leq 2 \ln n$. To show $f(\boldsymbol{x}) \leq 2 \ln n$ for all iterations t before terminate, it suffices to show that $f(\boldsymbol{x})$ is non-increasing during the process. From Lemma 6.3.2,

$$\begin{aligned} f(\boldsymbol{x}^{(t+1)}) - f(\boldsymbol{x}^{(t)}) &\leq \alpha\langle \boldsymbol{a}, (1+\Delta)\cdot\Delta\cdot\boldsymbol{x}^{(t)}\rangle - \alpha\langle \boldsymbol{b}, (1-\Delta)\cdot\Delta\cdot\boldsymbol{x}^{(t)}\rangle \\ &= \sum_i \alpha\Delta_i\boldsymbol{x}_i(\boldsymbol{a}_i(1+\Delta_i) - \boldsymbol{b}_i(1-\Delta_i)). \end{aligned}$$

For each $i \in [m]$, by our update rule (6.5), either Δ_i, or $\Delta_i = \frac{1}{2}(1 - \frac{\boldsymbol{a}_i}{\boldsymbol{b}_i})$, in which case

$$\boldsymbol{a}_i(1+\Delta_i) - \boldsymbol{b}_i(1-\Delta_i) = \frac{3\boldsymbol{a}_i\boldsymbol{b}_i - \boldsymbol{a}_i^2}{2\boldsymbol{b}_i} - \frac{\boldsymbol{a}_i + \boldsymbol{b}_i}{2} = \frac{2\boldsymbol{a}_i\boldsymbol{b}_i - \boldsymbol{a}_i^2 - \boldsymbol{b}_i^2}{2\boldsymbol{b}_i} \leq 0,$$

so all the summands are non-positive, thus $f(\boldsymbol{x})$ is non-increasing. □

The above lemma guarantees that the difference between $\operatorname{lmax}(\boldsymbol{P}\boldsymbol{x})$ and $\operatorname{lmin}(\boldsymbol{C}\boldsymbol{x})$ is bounded by $2 \ln n$, which by Eqn. (6.4) suggests $\max \boldsymbol{P}\boldsymbol{x} \leq \min \boldsymbol{C}\boldsymbol{x} + O(\ln n)$.Then when the two terms are large at termination, we are approximately feasible as the difference is a factor of ϵ smaller.

Lemma 6.3.4. *If Algorithm 16 terminates with line 14, then it returns an* $\boldsymbol{x} \geq 0$ *with* $0 < \max \boldsymbol{P}\boldsymbol{x} \leq (1+\epsilon)\min \boldsymbol{C}\boldsymbol{x}$.

Proof. Suppose the algorithm terminates at iteration T, that is, $\max \boldsymbol{P}\boldsymbol{x}^{(T)} \geq \frac{10 \ln n}{\epsilon}$ or $\min \boldsymbol{C}\boldsymbol{x}^{(T)} \geq \frac{10 \ln n}{\epsilon}$. Consider iteration $T-1$, the covering matrix is not empty (otherwise,

the algorithm terminates before iteration T). Since $\boldsymbol{x}^{(T)} = \boldsymbol{x}^{(T-1)} \cdot (1 + \alpha\Delta^{(T-1)}) \leq (1 + \frac{5\epsilon}{\ln n})\boldsymbol{x}^{(T-1)}$, we have $\max \boldsymbol{P}\boldsymbol{x}^{(T-1)} \geq \frac{5\ln n}{\epsilon}$ or $\min \boldsymbol{C}\boldsymbol{x}^{(T-1)} \geq \frac{5\ln n}{\epsilon}$.

By Lemma 6.3.3,

$$\max \boldsymbol{P}\boldsymbol{x}^{(T-1)} \leq \text{lmax}(\boldsymbol{P}\boldsymbol{x}^{(T-1)}) \leq \text{lmin}(\boldsymbol{C}^{(T-1)}\boldsymbol{x}^{(T-1)}) + 2\ln n \leq \min \boldsymbol{C}\boldsymbol{x}^{(T-1)} + 2\ln n.$$

Since $2\ln n \leq \epsilon \cdot \frac{5\ln n}{\epsilon}$, we have

$$\max \boldsymbol{P}\boldsymbol{x}^{(T-1)} \leq (1+\epsilon)\min \boldsymbol{C}\boldsymbol{x}^{(T-1)}.$$

This also gives $\max \boldsymbol{P}\boldsymbol{x}^{(T)} \leq (1+\epsilon)\min \boldsymbol{C}\boldsymbol{x}^{(T)}$, since $\boldsymbol{x}^{(T)}$ is within in multiplicative factor $1 + \frac{\epsilon}{10\ln n}$ of $\boldsymbol{x}^{(T-1)}$. Since we start with $\boldsymbol{x} > 0$, and only increase $\boldsymbol{x}$, we also have $\max \boldsymbol{P}\boldsymbol{x} > 0$. So the $\boldsymbol{x}$ we return at the end satisfies (6.1). □

6.3.2 Analysis of Convergence

So far we have proved that Algorithm 16 will terminate, and will either output $\boldsymbol{x}$ satisfying (6.1) at the end, or terminate earlier and correctly certify (6.2) is infeasible. In this section we show that if (6.2) is feasible, Algorithm 16 must finish with the first case in $\tilde{O}(\frac{1}{\epsilon^3})$ iterations, so if the algorithm takes more than $\frac{1000\ln n \ln(\frac{m}{\epsilon})}{\epsilon^3}$ iterations to complete, we can terminate it, and correctly output that (6.2) is infeasible.

We adapt the concept *phase* from Young's algorithm. Note phase is only used in our analysis, and our algorithm does not contain phase. Formally, phase s contains the iterations t such that

$$\frac{n_P}{n_C} \cdot 2^s \leq \frac{\mathbf{1}^T\boldsymbol{y}^{(t)}}{\mathbf{1}^T\boldsymbol{z}^{(t)}} < \frac{n_P}{n_C} \cdot 2^{s+1}$$

where n_P is the number of packing constraints and n_C is the number of covering constraints.

Since we only increase $\boldsymbol{x}$, $\frac{\mathbf{1}^T\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{z}}$ is monotonically increasing, so each phase covers a con-

secutive sequence of iterations. Furthermore, as $\ln(\frac{\mathbf{1}^T\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{z}}) = \operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}) + \operatorname{lmin}(\boldsymbol{C}\boldsymbol{x})$ measures global progress towards termination, each phase captures a fixed amount of progress. From our definition of phases, and the termination condition, we have

Lemma 6.3.5. *The total number of phases in Algorithm 16 is* $O(\frac{\log n}{\epsilon})$.

Proof. Since $\boldsymbol{x}$ is monotonically increasing, $\boldsymbol{y} = \exp(\boldsymbol{P}\boldsymbol{x})$ and $\boldsymbol{z} = \exp(-\boldsymbol{C}\boldsymbol{x})$ are monotonically increasing and decreasing respectively, which implies that the quantity $\frac{\mathbf{1}^T\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{z}}$ is monotonically increasing. Initially $\boldsymbol{P}\boldsymbol{x}^{(0)} \geq 0$, $\boldsymbol{C}\boldsymbol{x}^{(0)} \geq 0$, we know $\frac{\mathbf{1}^T\boldsymbol{y}^{(0)}}{\mathbf{1}^T\boldsymbol{z}^{(0)}} \geq \frac{n_P}{n_C}$. By the termination condition in Algorithm 16, the ratio never goes beyond $n_P \exp(\frac{10\log n}{\epsilon})$. Therefore, the total number of phases is $O(\frac{\log n}{\epsilon})$. □

We now bound the number of iterations in a single phase. The iterations of a phase are divided into two groups, the bad iterations and the good iterations, formally defined as follows.

Definition 6.3.6. *If in an iteration* t*, we have for all* i

$$\frac{\boldsymbol{a}_i^{(t)}}{\boldsymbol{b}_i^{(t)}} > \frac{1}{3}, \tag{6.6}$$

then we call it a good *iteration. Otherwise we call it a* bad *iteration.*

Note a phase may contain only bad iterations or only good iterations. We bound the total number of iterations in the two groups separately.

As discussed earlier, the bad iterations capture the initial warm-up iterations of a phase, where in any bad iteration, we can identify some variable $\boldsymbol{x}_i$ with a strong signal (i.e. $\frac{\boldsymbol{a}_i}{\boldsymbol{b}_i} \leq \frac{1}{3}$), so we can increase the variable by a lot. This restricts the warm-up sequence from getting too long, and we formalize the intuition in the following lemma.

Lemma 6.3.7. *In a single phase, the number of bad iterations is at most* $O(\ln n \ln(\frac{m}{\epsilon})/\epsilon)$.

Proof. We will prove the result by showing that there cannot be any bad iteration after the initial $100 \ln n \ln(\frac{m}{\epsilon})/\epsilon$ iterations of a phase. By contradiction, if for any variable i, after $\Omega(\ln n \ln(\frac{m}{\epsilon})/\epsilon)$ iterations of a phase, we have at iteration t such that for some i,

$$\frac{\boldsymbol{a}_i^{(t)}}{\boldsymbol{b}_i^{(t)}} = \frac{(\boldsymbol{P}^T \boldsymbol{y}^{(t)})_i}{\mathbf{1}^T \boldsymbol{y}^{(t)}} \frac{\mathbf{1}^T \boldsymbol{z}^{(t)}}{(\boldsymbol{C}^T \boldsymbol{z}^{(t)})_i} \leq \frac{1}{3},$$

then this ratio is at most $\frac{2}{3}$ in all previous iterations of this phase, since $\frac{\left(\boldsymbol{P}^T \boldsymbol{y}\right)_i}{\left(\boldsymbol{C}^T \boldsymbol{z}\right)_i}$ is monotonically increasing, and $2^s \leq \frac{\mathbf{1}^T \boldsymbol{y}}{\mathbf{1}^T \boldsymbol{z}} < 2^{s+1}$ in this phase. Equivalently, this is saying $\boldsymbol{a}_i \leq \frac{2}{3} \boldsymbol{b}_i$, so $i \in B$ in all previous $\Omega(\ln n \ln(\frac{m}{\epsilon})/\epsilon)$ iterations of the phase, and $\Delta_i \geq \frac{1}{6}$ in all those iterations.

Each iteration the multiplicative update on $\boldsymbol{x}_i$ is $(1+\alpha\Delta_i)$, which is $(1+\Theta(\frac{\epsilon}{10 \ln n}))$ since $\Delta_i \geq \frac{1}{6}$. As $\boldsymbol{x}_i$ starts with $\frac{1}{m\|P_{:,i}\|_\infty}$, after $100 \ln n \ln(\frac{m}{\epsilon})/\epsilon$ updates, we have $\boldsymbol{x}_i >> \frac{10 \ln n}{\epsilon \|P_{:,i}\|_\infty}$, which gives $\max \boldsymbol{P}\boldsymbol{x} >> \frac{10 \ln n}{\epsilon}$, so the algorithm must have terminated. □

The above lemma guarantees that all iterations after the first $100 \ln n \ln(\frac{m}{\epsilon})/\epsilon$ must be good iterations, so we proceed to bound the number of these good iterations in a single phase. Without loss of generality, we index these good iterations in a phase as $1, \ldots, T$ by shifting t.

We first identify one variable that must be updated extensively in these iterations.

Claim 6.3.8. *Suppose the instance* (6.2) *is feasible, then There exists* $i \in [m]$ *such that*

$$\sum_{t=1}^{T} \boldsymbol{b}_i^{(t)} - \boldsymbol{a}_i^{(t)} \geq 10\epsilon \sum_{t=1}^{T} \boldsymbol{b}_i^{(t)}. \tag{6.7}$$

Proof. Define $\overline{\boldsymbol{y}}$ and $\overline{\boldsymbol{z}}$ to be the sum of the normalized gradients of iterations $1, \ldots, T$, that is,

$$\overline{\boldsymbol{y}} = \sum_{t=1}^{T} \frac{\boldsymbol{y}^{(t)}}{\mathbf{1}^T \boldsymbol{y}^{(t)}}, \quad \overline{\boldsymbol{z}} = \sum_{t=1}^{T} \frac{\boldsymbol{z}^{(t)}}{\mathbf{1}^T \boldsymbol{z}^{(t)}}.$$

Note $\mathbf{1}^T\overline{\boldsymbol{y}} = \mathbf{1}^T\overline{\boldsymbol{z}} = T$. Recall $\boldsymbol{a}_i^{(t)}$ and $\boldsymbol{b}_i^{(t)}$ are respectively $\frac{(\boldsymbol{P}^T\boldsymbol{y}^{(t)})_i}{\mathbf{1}^T\boldsymbol{y}^{(t)}}$ and $\frac{(\boldsymbol{C}^T\boldsymbol{z}^{(t)})_i}{\mathbf{1}^T\boldsymbol{z}^{(t)}}$, then

$$\sum_{t=1}^{T} \boldsymbol{a}_i^{(t)} = \frac{T(\boldsymbol{P}^T\overline{\boldsymbol{y}})_i}{\mathbf{1}^T\overline{\boldsymbol{y}}}, \quad \sum_{t=1}^{T} \boldsymbol{b}_i^{(t)} = \frac{T(\boldsymbol{C}^T\overline{\boldsymbol{z}})_i}{\mathbf{1}^T\overline{\boldsymbol{z}}}.$$

Assume by contradiction, $\forall\, i \in [m]$, $\sum_{t=1}^{T} \boldsymbol{a}_i^{(t)} > (1-10\epsilon)\sum_{t=1}^{T} \boldsymbol{b}_i^{(t)}$, that is,

$$\frac{\boldsymbol{P}^T\overline{\boldsymbol{y}}}{\mathbf{1}^T\overline{\boldsymbol{y}}} > (1-10\epsilon)\frac{\boldsymbol{C}^T\overline{\boldsymbol{z}}}{\mathbf{1}^T\overline{\boldsymbol{z}}}.$$

By Lemma 6.1.1, $\overline{\boldsymbol{y}}, \overline{\boldsymbol{z}}$ certify infeasibility of the instance (6.2), which contradicts the assumption. □

The above claim gives us a variable that on average has smaller packing gradients than covering gradients in this iteration. Together with the property we have on the good iterations (6.6), we can bound the number of good iterations.

Lemma 6.3.9. *In a single phase, the number of good iterations is at most* $O(\ln n \ln(\frac{m}{\epsilon})/\epsilon^2)$.

Proof. Let $\boldsymbol{x}_i$ be a variable satisfying Eqn. (6.7). We want to turn Eqn. (6.7) into some lower bound on the total multiplicative update on $\boldsymbol{x}_i$ through these iterations. Intuitively, a bad case is that in some iteration t, $\boldsymbol{a}_i^{(t)}, \boldsymbol{b}_i^{(t)}$ are much larger than the values in other iterations, since they can dominate the terms from other iterations in Eqn. (6.7), but not much to the total update of $\boldsymbol{x}_i$, since their ratio is what matters to the update. However, since we are inside one single phase, and only looking at good iterations, we can show the bad scenario will not show up.

Formally, let $l = \boldsymbol{a}_i^{(1)}$ and $u = \boldsymbol{b}_i^{(1)}$. Since $(\boldsymbol{P}^T\boldsymbol{y})_i$ monotonically increases, and $\mathbf{1}^T\boldsymbol{y}$ will increase but not by more than a factor of 2 in a phase, we have

$$\boldsymbol{a}_i^{(t)} = \frac{(\boldsymbol{P}^T\boldsymbol{y}^{(t)})_i}{\mathbf{1}^T\boldsymbol{y}^{(t)}} \geq l/2 \qquad \forall t = 1, \ldots, T \tag{6.8}$$

Similarly, we have

$$\boldsymbol{b}_i^{(t)} = \frac{(\boldsymbol{C}^T \boldsymbol{z}^{(t)})_i}{\mathbf{1}^T \boldsymbol{z}^{(t)}} \leq 2u \qquad \forall t = 1, \ldots, T \tag{6.9}$$

Furthermore, since we are looking at the good iterations, we have

$$l \geq \frac{1}{3}u.$$

The inequalities above allow us to turn the difference-based guarantee from Eqn. (6.7) into lower bounds on ratios we need.

By the update (6.5), we have

$$\Delta_i^{(t)} \geq \frac{(1 - \frac{\epsilon}{50})\boldsymbol{b}_i^{(t)} - \boldsymbol{a}_i^{(t)}}{2\boldsymbol{b}_i^{(t)}}.$$

So we can lower bound the total update on $\boldsymbol{x}_i$ as follows

$$\begin{aligned}
\boldsymbol{x}_i^{(T)} \geq & \boldsymbol{x}_i^{(1)} \exp\left(\frac{\alpha \sum_t \Delta_i^{(t)}}{2}\right) \\
= & \boldsymbol{x}_i^{(1)} \exp\left(\frac{\alpha}{4} \sum_t \frac{(1 - \frac{\epsilon}{50})\boldsymbol{b}_i^{(t)} - \boldsymbol{a}_i^{(t)}}{\boldsymbol{b}_i^{(t)}}\right) \\
\geq & \boldsymbol{x}_i^{(1)} \exp\left(\frac{\alpha}{4} \sum_t \left(\frac{\boldsymbol{b}_i^{(t)} - \boldsymbol{a}_i^{(t)}}{2u} - \frac{\epsilon}{50}\right)\right)
\end{aligned}$$

where we used (6.9) in the last line.

From Eqn. (6.7), we have

$$\begin{aligned}
\sum_t \boldsymbol{b}_i^{(t)} - \boldsymbol{a}_i^{(t)} \geq & 10\epsilon \sum_i \boldsymbol{b}_i^{(t)} \\
\geq & \frac{10\epsilon}{1 - 10\epsilon} \sum_t \boldsymbol{a}_i^{(t)} \\
\geq & \frac{\epsilon u T}{1 - 10\epsilon} \geq \epsilon u T.
\end{aligned}$$

The first two lines both follow from Eqn. (6.7), the next line follows from $\boldsymbol{a}_i^{(t)} \geq l/2 \geq u/6$.

Thus

$$\boldsymbol{x}_i^{(T)} \geq \boldsymbol{x}_i^{(1)} \exp\left(\frac{\epsilon\alpha T}{8} - \frac{\epsilon\alpha T}{200}\right) \geq \frac{1}{m\left\|\boldsymbol{P}_{:,i}\right\|_\infty} \exp\left(\frac{\epsilon\alpha T}{10}\right).$$

If $T \geq \frac{100 \ln n \ln \frac{m}{\epsilon}}{\epsilon^2} \geq \frac{100 \ln \frac{m}{\epsilon}}{\epsilon\alpha}$, we have $\boldsymbol{x}_i^{(T)} >> \frac{10 \ln n}{\epsilon\|\boldsymbol{P}_{:,i}\|_\infty}$. So the algorithm must have terminated since $\max \boldsymbol{P}\boldsymbol{x} >> \frac{10 \ln n}{\epsilon}$. □

Lemma 6.3.7 and Lemma 6.3.9 bound the total number of iterations in a phase by $\tilde{O}(\frac{1}{\epsilon^2})$, together with the bound on the number of phases, which is $\tilde{O}(\frac{1}{\epsilon})$, we guarantee the total number of iterations in Algorithm 16 is $\tilde{O}(\frac{1}{\epsilon^3})$ if the LP in (6.2) is feasible.

Proof of Theorem 1.1.6. The correctness and convergence follows from the lemmas in the prior sections. We only need to look at the running time and total work.

At each iteration, we compute all updated values in $O(\log N)$ parallel time. Since the total number of iterations is $\tilde{O}(\frac{1}{\epsilon^3})$, the algorithm terminates in parallel time $\tilde{O}(\frac{1}{\epsilon^3})$.

To see the total work, consider the following implementation. For each $i \in [m]$, we maintain $\boldsymbol{P}_{ji}\boldsymbol{x}_i$ if $\boldsymbol{P}_{ij} \neq 0$; similarly we maintain $\boldsymbol{C}_{ji}\boldsymbol{x}_i$ if $\boldsymbol{C}_{ij} \neq 0$. Besides, we maintain the values of $\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{P}^T\boldsymbol{y}, \boldsymbol{C}^T\boldsymbol{z}, \mathbf{1}^T\boldsymbol{y}$ and $\mathbf{1}^T\boldsymbol{z}$. When we update $\boldsymbol{x}_i$, we update these values accordingly, with work proportional to the number of non-zero entries in the ith column of the constraint matrix. For each fixed variable $\boldsymbol{x}_i$, the total time of updates is at most $\tilde{O}(\frac{1}{\epsilon^2})$. Thus, the work on this part is $\tilde{O}(\frac{N}{\epsilon^2})$. Additionally, we need to compute the $\boldsymbol{a}_i, \boldsymbol{b}_i$ for all variables at the beginning of each iteration to determine which variables to update, this takes $\tilde{O}(N)$ work each iteration, so the total work is $\tilde{O}(\frac{N}{\epsilon^3})$.

We see the majority of the work is actually on computing the gradients for the variables we may not update. We point out that we can implement the same lazy update as in [78], which on a high level is just that if a variable has a large $\frac{\boldsymbol{a}_i}{\boldsymbol{b}_i}$ in an iteration, and is not updated, we don't recompute its gradients, until $\frac{\mathbf{1}^T\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{z}}$ grows by more than a factor of $1+\epsilon$. This can reduce the work to $\tilde{O}(\frac{N}{\epsilon^2})$, but requires a centralized step to control the phases. We omit the details as it is a straightforward adaptation. □

6.3.3 Pure Packing and Pure Covering LPs

We point out that in the case of pure packing or pure covering LPs, Algorithm 16 converges in $\tilde{O}(\frac{1}{\epsilon^2})$ iterations. This improves upon the result of [61], since our algorithm is deterministic, and does not need centralized steps.

Given packing LP in the optimization form

$$\max_{\boldsymbol{x} \geq 0} \{\mathbf{1}^T \boldsymbol{x} : \boldsymbol{P}\boldsymbol{x} \leq \mathbf{1}\}$$

via standard reduction and scaling, we need to solve a $(1+\epsilon)$-feasibility problem the same as in the mixed packing and covering case specified in (6.1) and (6.2). In the case of pure packing, we will have $\boldsymbol{C} = c\mathbf{1}^T$ for some constant c.

The correctness proof follows from the mixed case, and we discuss how we get faster convergence for pure packing.

The special structure of $\boldsymbol{C}$ greatly simplifies the convergence analysis, as now $\boldsymbol{z}$ is a scalar, and $\boldsymbol{b}_i^{(t)} = c$ for all variables $\boldsymbol{x}_i$ across all iterations t. This allows us to aggregate the good iterations from all phases, and bound the total number of good phases by $\tilde{O}(\frac{1}{\epsilon^2})$, which will lead to $\tilde{O}(\frac{1}{\epsilon^2})$ total iterations.

In particular, now we look at all the good iterations across all phases together, and WLOG number them $1, \ldots, T$. Claim 6.3.8 still holds, as it does not rely on phases. Then we can prove a stronger version of Lemma 6.3.9

Lemma 6.3.10. *The total number of good iterations T is at most $O(\ln n \ln(\frac{m}{\epsilon})/\epsilon^2)$.*

The proof is a straightforward adaptation of the proof of Lemma 6.3.9. The proof is simpler, since now $\boldsymbol{b}_i$ is a constant across all iterations, so the property (6.6) we have on the good iterations directly put all the values on the same scale, so we can lower bound the total update on $\boldsymbol{x}_i$ across all good iterations.

Appendices

APPENDIX A

PROOFS OF LINEAR ALGEBRA FACTS

A.1 Schur Complements

Proof of Fact 2.1.2. We expand the left hand side,

$$\begin{pmatrix} \boldsymbol{x}^\top & \boldsymbol{y}^\top \end{pmatrix} \boldsymbol{A} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \boldsymbol{x}^\top \boldsymbol{A}_{SS} \boldsymbol{x} + 2\boldsymbol{x}^\top \boldsymbol{A}_{ST} \boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{A}_{TT} \boldsymbol{y}. \tag{A.1}$$

Taking derivative w.r.t. $\boldsymbol{y}$ and setting it to be 0 give that

$$2\boldsymbol{A}_{TT}\boldsymbol{y} + 2\boldsymbol{A}_{ST}^\top \boldsymbol{x} = \boldsymbol{0}.$$

Plugging $\boldsymbol{y} = -\boldsymbol{A}_{TT}^\dagger \boldsymbol{A}_{ST}^\top \boldsymbol{x}$ into (A.1),

$$\min_{\boldsymbol{y}} \begin{pmatrix} \boldsymbol{x}^\top & \boldsymbol{y}^\top \end{pmatrix} \boldsymbol{A} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \boldsymbol{x}^\top \boldsymbol{A}_{SS} \boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{A}_{ST} \boldsymbol{A}_{TT}^\dagger \boldsymbol{A}_{ST}^\top \boldsymbol{x} = \boldsymbol{x}^\top \textsc{Sc}[\boldsymbol{A}]_T \boldsymbol{x}.$$

This completes the proof. □

To prove Fact 2.1.3, we need the following special case of Weyl inequalities.

Theorem A.1.1 (A special case of Weyl inequalities)**.** *Let $\boldsymbol{H} \in \mathbb{R}^n$ and $\boldsymbol{H} = \boldsymbol{H}_1 + \boldsymbol{H}_2$ where $\boldsymbol{H}_1, \boldsymbol{H}_2$ are symmetric matrices and $\boldsymbol{H}_2$ is a PSD matrix. Let $\lambda_1(\cdot) \geq \ldots \geq \lambda_n(\cdot)$ be eigenvalues of a matrix. Then, $\lambda_i(\boldsymbol{H}) \geq \lambda_i(\boldsymbol{H}_1), i = 1, 2, \ldots, n$.*

Proof of Fact 2.1.3. The first statement immediately follows Fact 2.1.2.

To prove the second statement, we decompose $\boldsymbol{A}$:

$$\begin{aligned}\boldsymbol{A} &= \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \textsc{Sc}[\boldsymbol{A}]_T \end{pmatrix} + \begin{pmatrix} \boldsymbol{A}_{SS} & \boldsymbol{A}_{ST} \\ \boldsymbol{A}_{ST}^\top & \boldsymbol{A}_{ST}^\top \boldsymbol{A}_{SS}^{-1} \boldsymbol{A}_{ST} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \textsc{Sc}[\boldsymbol{A}]_T \end{pmatrix} + \begin{pmatrix} \boldsymbol{A}_{SS}^{1/2} \\ \boldsymbol{A}_{ST}^\top \boldsymbol{A}_{SS}^{-1/2} \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{A}_{SS}^{1/2} \\ \boldsymbol{A}_{ST}^\top \boldsymbol{A}_{SS}^{-1/2} \end{pmatrix}.\end{aligned}$$

Here we assume that $\boldsymbol{A}_{SS}$ is invertible, otherwise we can use pseudo-inverse. The first matrix is symmetric, and the second matrix is symmetric and PSD. By Theorem A.1.1, $\lambda_{\max}(\boldsymbol{A}) \geq \lambda_{\max}(\textsc{Sc}[\boldsymbol{A}]_T)$. □

A.2 Solutions of Linear Systems

In this section, we prove some facts on linear system solutions.

Proof of Fact 2.2.4. First observe

$$\begin{aligned}\left\|\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top \boldsymbol{c}\right\|_{(\boldsymbol{A}^\top \boldsymbol{A})^\dagger}^2 &= (\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top \boldsymbol{c})^\top (\boldsymbol{A}^\top \boldsymbol{A})^\dagger (\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top \boldsymbol{c}) \\ &= (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{c})^\top \boldsymbol{\Pi}_{\boldsymbol{A}} (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{c}) \\ &= (\boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{A}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{c})^\top (\boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{A}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{c}) \\ &= \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{c}\|_2^2 .\end{aligned}$$

Taking $\boldsymbol{x} = \mathbf{0}$ it follows that $\|\boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{c}\| = \left\|\boldsymbol{A}^\top \boldsymbol{c}\right\|_{(\boldsymbol{A}^\top \boldsymbol{A})^\dagger}$, and combining our observations immediately gives the second part of the Fact.

A similar argument gives claims for $\|\boldsymbol{x} - \boldsymbol{x}^*\|_{\boldsymbol{A}^\top \boldsymbol{A}}$.

$$\begin{aligned}
\|\boldsymbol{x} - \boldsymbol{x}^*\|_{\boldsymbol{A}^\top \boldsymbol{A}}^2 &= (\boldsymbol{x} - \boldsymbol{x}^*)^\top \boldsymbol{A}^\top \boldsymbol{A} (\boldsymbol{x} - \boldsymbol{x}^*) \\
&= \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{x} - 2\boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{x}^* + (\boldsymbol{x}^*)^\top \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{x}^* \\
&= \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{x} - 2\boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{c} + \boldsymbol{c}^\top \boldsymbol{\Pi}_{\boldsymbol{A}}^\top \boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{c} \\
&= \|\boldsymbol{A} \boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{c}\|_2^2 .
\end{aligned}$$

The third equality uses the fact that $\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{c}$. This completes the proof. □

Proof of Lemma 2.2.5. As $\|\boldsymbol{A}\boldsymbol{c}\|_2^2$ is integral, the condition of $\boldsymbol{A}^\top \boldsymbol{c} \neq 0$ implies $\left\|\boldsymbol{A}^\top \boldsymbol{c}\right\|_2 \geq 1$. Consider the SVD of $\boldsymbol{A}$:

$$\boldsymbol{A} = \sum_i \sigma_i \boldsymbol{r}^i (\boldsymbol{s}^i)^\top .$$

Substituting it in for $\boldsymbol{A}^\top$ gives

$$\boldsymbol{A}^\top \boldsymbol{c} = \sum_i \left(\sigma_i (\boldsymbol{r}^i)^\top \boldsymbol{c}\right) \boldsymbol{s}^i,$$

and we will use $\alpha_i = \sigma_i (\boldsymbol{r}^i)^\top \boldsymbol{c}$ to denote the coefficients of $\boldsymbol{c}$ against the singular vectors of $\boldsymbol{A}$. Note that $\sigma_i = 0$ implies $\alpha_i = 0$. The norm condition on $\boldsymbol{A}^\top \boldsymbol{c}$ also means $\sum_i \alpha_i^2 = \sum_{i:\sigma_i \neq 0} \alpha_i^2 \geq 1$, which then gives

$$\begin{aligned}
\|\boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{c}\|_2^2 &= \boldsymbol{c}^\top \boldsymbol{A} \left(\boldsymbol{A}^\top \boldsymbol{A}\right)^\dagger \boldsymbol{A}^\top \boldsymbol{c} \\
&= \left(\sum_i \alpha_i (\boldsymbol{s}^i)^\top\right) \left(\sum_{i:\sigma_i \neq 0} \frac{1}{\sigma_i^2} \boldsymbol{s}^i (\boldsymbol{s}^i)^\top\right) \left(\sum_i \alpha_i \boldsymbol{s}^i\right) = \sum_{i:\sigma_i \neq 0} \frac{\alpha_i^2}{\sigma_i^2} \geq \frac{1}{\sigma_{\max}^2(\boldsymbol{A})}.
\end{aligned}$$

This completes the proof. □

Proof of Claim 3.1.2. The first inequality is trivial.

The second inequality follows the relations of norms,

$$\|\boldsymbol{c}\|_2 \leq \sqrt{n}\,\|\boldsymbol{c}\|_\infty \leq \sqrt{s}U.$$

For the third one, by the definition of Frobenius norm,

$$\|\boldsymbol{A}\|_F^2 = \sum_{i,j} \boldsymbol{A}_{ij}^2 = \sum_{i=1}^{\min\{m,n\}} \lambda_i(\boldsymbol{A}^\top \boldsymbol{A}).$$

Thus,

$$\lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A}) \leq \|\boldsymbol{A}\|_F^2 \leq sU^2,$$

and

$$\lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A}) \geq \frac{\|\boldsymbol{A}\|_F^2}{\min\{m,n\}} \geq \frac{1}{sU^2}.$$

By the definition of $\kappa(\boldsymbol{A}^\top \boldsymbol{A})$,

$$\lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A}) = \frac{\lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A})}{\kappa(\boldsymbol{A}^\top \boldsymbol{A})} \geq \frac{1}{sK^2U^2}.$$

This completes the proof. □

REFERENCES

[1] N. Megiddo, "On solving the linear programming problem approximately," *Contemporary mathematics*, vol. 114, pp. 35–50, 1990.

[2] J. Renegar, "Incorporating condition measures into the complexity theory of linear programming," *SIAM Journal on Optimization*, vol. 5, no. 3, pp. 506–524, 1995.

[3] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, ACM, 1984, pp. 302–311.

[4] Y. T. Lee and A. Sidford, "Path finding methods for linear programming: solving linear programs in $\tilde{O}(\sqrt{rank})$ iterations and faster algorithms for maximum flow," in *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, Available at http://arxiv.org/abs/1312.6677 and http://arxiv.org/abs/1312.6713, IEEE, 2014, pp. 424–433.

[5] L. G. Khachiyan, "A polynomial algorithm in linear programming," in *Doklady Academii Nauk SSSR*, vol. 244, 1979, pp. 1093–1096.

[6] V. Klee and G. J. Minty, "How good is the simplex algorithm," WASHINGTON UNIV SEATTLE DEPT OF MATHEMATICS, Tech. Rep., 1970.

[7] D. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time," in *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, ACM, 2001, pp. 296–305.

[8] D. A. Spielman, "Nsf award 1562041: generalized algebraic graph theory: algorithms and analysis," *ALGORITHMIC FOUNDATIONS*, 2016.

[9] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, ser. STOC '04, Chicago, IL, USA: ACM, 2004, pp. 81–90, ISBN: 1-58113-852-0.

[10] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman, "Sparsified cholesky and multigrid solvers for connection laplacians," in *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '16, Cambridge, MA, USA: ACM, 2016, pp. 842–850, ISBN: 978-1-4503-4132-5.

[11] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. B. Rao, A. Sidford, and A. Vladu, "Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2017, Montreal, Canada: ACM, 2017, pp. 410–419, ISBN: 978-1-4503-4528-6.

[12] M. B. Cohen, J. Kelner, R. Kyng, J. Peebles, R. Peng, A. B. Rao, and A. Sidford, "Solving directed laplacians in nearly linear time through sparse lu factorizations," in *Foundations of Computer Science (FOCS), 2018 IEEE 50th Annual Symposium on*, IEEE, 2018.

[13] N. E. Young, "Sequential and parallel algorithms for mixed packing and covering," in *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, 2001, pp. 538–546.

[14] V. Strassen, "Gaussian elimination is not optimal," *Numerische mathematik*, vol. 13, no. 4, pp. 354–356, 1969.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[16] V. V. Williams, "Multiplying matrices faster than coppersmith-winograd," in *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '12, New York, New York, USA: ACM, 2012, pp. 887–898, ISBN: 978-1-4503-1245-5.

[17] F. Le Gall, "Powers of tensors and fast matrix multiplication," in *Proceedings of the 39th international symposium on symbolic and algebraic computation*, Available at: https://arxiv.org/abs/1401.7714, ACM, 2014, pp. 296–303.

[18] R. Raz, "On the complexity of matrix product," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, ACM, 2002, pp. 144–151.

[19] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, 1. NBS Washington, DC, 1952, vol. 49.

[20] C. Musco, C. Musco, and A. Sidford, "Stability of the lanczos method for matrix function approximation," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, Society for Industrial and Applied Mathematics, 2018, pp. 1605–1624.

[21] A. Madry, "Navigating central path with electrical flows: from flows to matchings, and back," in *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, ser. FOCS '13, Washington, DC, USA: IEEE Computer Society, 2013, pp. 253–262, ISBN: 978-0-7695-5135-7.

[22] ——, "Computing maximum flow with augmenting electrical flows," in *Proceedings of the 2016 IEEE 57th Annual Symposium on Foundations of Computer Science*, ser. FOCS '16, Washington, DC, USA: IEEE Computer Society, 2016.

[23] M. B. Cohen, A. Madry, P. Sankowski, and A. Vladu, "Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{O}(M^{10/7} \log W)$ time: (extended abstract)," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '17, Barcelona, Spain: Society for Industrial and Applied Mathematics, 2017, pp. 752–771.

[24] S. I. Daitch and D. A. Spielman, "Faster approximate lossy generalized flow via interior point algorithms," in *Proceedings of the 40th annual ACM symposium on Theory of computing*, ser. STOC '08, Available at http://arxiv.org/abs/0803.0988, Victoria, British Columbia, Canada: ACM, 2008, pp. 451–460, ISBN: 978-1-60558-047-0.

[25] S.-H. Teng, "The Laplacian Paradigm: Emerging Algorithms for Massive Graphs," in *Theory and Applications of Models of Computation*, 2010, pp. 2–14.

[26] I. Koutis, G. L. Miller, and R. Peng, "Approaching optimality for solving SDD linear systems," in *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, ser. FOCS '10, Available at http://arxiv.org/abs/1003.2958, Washington, DC, USA: IEEE Computer Society, 2010, pp. 235–244, ISBN: 978-0-7695-4244-7.

[27] ——, "A nearly-m log n time solver for SDD linear systems," in *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, ser. FOCS '11, Available at http://arxiv.org/abs/1102.4842, Washington, DC, USA: IEEE Computer Society, 2011, pp. 590–598, ISBN: 978-0-7695-4571-4.

[28] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu, "A simple, combinatorial algorithm for solving sdd systems in nearly-linear time," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, ACM, 2013, pp. 911–920.

[29] R. Peng and D. A. Spielman, "An efficient parallel solver for sdd linear systems," in *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, ACM, 2014, pp. 333–342.

[30] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu, "Solving sdd linear systems in nearly m log 1/2 n time," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, ACM, 2014, pp. 343–352.

[31] R. Kyng and S. Sachdeva, "Approximate gaussian elimination for laplacians-fast, sparse, and simple," in *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, IEEE, 2016, pp. 573–582.

[32] H. H. Chin, A. Madry, G. L. Miller, and R. Peng, "Runtime guarantees for regression problems," in *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, ser. ITCS '13, Berkeley, California, USA: ACM, 2013, pp. 269–282, ISBN: 978-1-4503-1859-4.

[33] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1068–1080, 2008.

[34] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A new alternating minimization algorithm for total variation image reconstruction," *SIAM Journal on Imaging Sciences*, vol. 1, no. 3, pp. 248–272, 2008.

[35] M. Arie-Nachimson, S. Z. Kovalsky, I. Kemelmacher-Shlizerman, A. Singer, and R. Basri, "Global motion estimation from point matches," in *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, IEEE, 2012, pp. 81–88.

[36] O. Ozyesil, A. Singer, and R. Basri, "Stable camera motion estimation using convex programming," *SIAM Journal on Imaging Sciences*, vol. 8, no. 2, pp. 1220–1262, 2015.

[37] M. B. Cohen, Y. T. Lee, G. Miller, J. Pachocki, and A. Sidford, "Geometric median in nearly linear time," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, Available at: https://arxiv.org/abs/1606.05225, ACM, 2016, pp. 9–21.

[38] A. Singer and Y. Shkolnisky, "Three-dimensional structure determination from common lines in cryo-EM by eigenvectors and semidefinite programming," *SIAM journal on imaging sciences*, vol. 4, no. 2, pp. 543–572, 2011.

[39] Y. Shkolnisky and A. Singer, "Viewing direction estimation in cryo-EM using synchronization," *SIAM journal on imaging sciences*, vol. 5, no. 3, pp. 1088–1110, 2012.

[40] Z. Zhao and A. Singer, "Rotationally invariant image representation for viewing direction classification in cryo-EM," *Journal of structural biology*, vol. 186, no. 1, pp. 153–166, 2014.

[41] B. Alexeev, A. S. Bandeira, M. Fickus, and D. G. Mixon, "Phase retrieval with polarization," *SIAM Journal on Imaging Sciences*, vol. 7, no. 1, pp. 35–66, 2014.

[42] S. Marchesini, Y.-C. Tu, and H.-t. Wu, "Alternating projection, ptychographic imaging and phase synchronization," *arXiv preprint arXiv:1402.0550*, 2014.

[43] G. Shklarski and S. Toledo, "Rigidity in finite-element matrices: sufficient conditions for the rigidity of structures and substructures," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 1, pp. 7–40, 2008.

[44] S. I. Daitch and D. A. Spielman, "Support-graph preconditioners for 2-dimensional trusses," *arXiv preprint cs/0703119*, 2007.

[45] A. Singer and H.-T. Wu, "Vector diffusion maps and the connection laplacian," *Communications on pure and applied mathematics*, vol. 65, no. 8, pp. 1067–1144, 2012.

[46] A. S. Bandeira, A. Singer, and D. A. Spielman, "A cheeger inequality for the graph connection laplacian," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 4, pp. 1611–1630, 2013.

[47] A. George, "Nested dissection of a regular finite element mesh," *SIAM Journal on Numerical Analysis*, vol. 10, no. 2, pp. 345–363, 1973.

[48] R. J. Lipton, D. J. Rose, and R. E. Tarjan, "Generalized nested dissection," *SIAM journal on numerical analysis*, vol. 16, no. 2, pp. 346–358, 1979.

[49] G. L. Miller and W. Thurston, "Separators in two and three dimensions," in *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, ACM, 1990, pp. 300–309.

[50] N. Alon and R. Yuster, "Solving linear systems through nested dissection," in *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, IEEE, 2010, pp. 225–234.

[51] M. Luby and N. Nisan, "A parallel approximation algorithm for positive linear programming," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, 1993, pp. 448–457.

[52] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: a meta-algorithm and applications.," *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012.

[53] Z. Allen-Zhu and L. Orecchia, "Nearly-linear time positive LP solver with faster convergence rate," in *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, ser. STOC '15, Newer version available at `http://arxiv.org/abs/1411.1124`, 2015, pp. 229–236.

[54] ——, "Using optimization to break the epsilon barrier: a faster and simpler width-independent algorithm for solving positive linear programs in parallel," in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '15, 2015, pp. 1439–1456.

[55] D. Wang, S. Rao, and M. W. Mahoney, "Unified acceleration method for packing and covering problems via diameter reduction," in *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, 2016, 50:1–50:13.

[56] R. Kyng and P. Zhang, "Hardness results for structured linear systems," in *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, IEEE, 2017, pp. 684–695.

[57] R. Kyng, R. Peng, D. Wang, and P. Zhang, "Packing lps are hard to solve accurately, assuming linear equations are hard," 2018, in submission.

[58] L. Trevisan and F. Xhafa, "The parallel complexity of positive linear programming," *Parallel Processing Letters*, vol. 8, no. 04, pp. 527–533, 1998.

[59] R. Kyng, R. Peng, R. Schwieterman, and P. Zhang, "Incomplete nested dissection," in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, ACM, 2018, pp. 404–417.

[60] M. W. Mahoney, S. Rao, D. Wang, and P. Zhang, "Approximating the solution to mixed packing and covering lps in parallel $\widetilde{O}(\epsilon^{-3})$ time," in *LIPIcs-Leibniz International Proceedings in Informatics*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, vol. 55, 2016.

[61] D. Wang, M. W. Mahoney, N. Mohan, and S. Rao, "Faster parallel solver for positive linear programs via dynamically-bucketed selective coordinate descent," *CoRR*, vol. abs/1511.06468, 2015.

[62] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997, vol. 50.

[63] K. L. Clarkson and D. P. Woodruff, "Low-rank approximation and regression in input sparsity time," *Journal of the ACM (JACM)*, vol. 63, no. 6, p. 54, 2017.

[64] E. G. Boman, D. Chen, O. Parekh, and S. Toledo, "On factor width and symmetric h-matrices," *Linear algebra and its applications*, vol. 405, pp. 239–248, 2005.

[65] D. Goldfarb and W. Yin, "Second-order cone programming methods for total variation-based image restoration," *SIAM Journal on Scientific Computing*, vol. 27, no. 2, pp. 622–645, 2005.

[66] M. B. Cohen, B. T. Fasy, G. L. Miller, A. Nayyeri, R. Peng, and N. Walkington, "Solving 1-laplacians in nearly linear time: collapsing and expanding a topological ball," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, Available

at: https://www.cs.cmu.edu/~glmiller/Publications/Papers/CoFMNPW14.pdf, 2014, pp. 204–216.

[67] G. Barequet and S. Har-Peled, "Efficiently approximating the minimum-volume bounding box of a point set in three dimensions," *J. Algorithms*, vol. 38, no. 1, pp. 91–109, Jan. 2001.

[68] O. Axelsson, "A survey of preconditioned iterative methods for linear systems of algebraic equations," *BIT Numerical Mathematics*, vol. 25, no. 1, pp. 165–187, 1985.

[69] D. A. Spielman, "Spectral graph theory and its applications," in *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, IEEE, 2007, pp. 29–38.

[70] S. J. Wright, *Primal-dual interior-point methods*. SIAM, 1997.

[71] Y. Ye, *Interior point algorithms: theory and analysis*. John Wiley & Sons, 2011, vol. 44, Available at: http://web.stanford.edu/~yyye/main.ps.

[72] A. Nemirovski, "Interior point polynomial time methods in convex programming," *Lecture notes*, 2004.

[73] S. Boyd and L. Vandenberghe, *Convex Optimization*. Camebridge University Press, 2004, Available at https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf.

[74] Y. T. Lee and A. Sidford, "Efficient inverse maintenance and faster algorithms for linear programming," in *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, Available at: https://arxiv.org/abs/1503.01752, IEEE, 2015, pp. 230–249.

[75] A. Madry, "Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms," in *STOC '10: Proceedings of the 42nd ACM symposium on Theory of computing*, Cambridge, Massachusetts, USA: ACM, 2010, pp. 121–130, ISBN: 978-1-4503-0050-6.

[76] D. Goldfarb and W. Yin, "Second-order cone programming methods for total variation-based image restoration," *SIAM J. Sci. Comput*, vol. 27, pp. 622–645, 2004.

[77] G. N. Frederickson, "Fast algorithms for shortest paths in planar graphs, with applications," *SIAM J. Comput.*, vol. 16, no. 6, pp. 1004–1022, 1987.

[78] N. E. Young, "Nearly linear-time approximation schemes for mixed packing/covering and facility-location linear programs," *CoRR*, vol. abs/1407.3015, 2014.